HYPERQUANT: A NEAR-ENTROPY-OPTIMAL LATTICE QUANTIZATION PIPELINE FOR LARGE LANGUAGE AND DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Data-free post-training quantization has converged on rotation plus a fixed-rate codebook: whiten a weight tile toward Gaussianity, then round it to an MSE-optimal grid. That last step spends $\log_2 N$ bits per index however unevenly the indices are used.

We introduce HYPERQUANT (Hadamard, optimallY Packing, Entropy Rice-coding), which replaces that codebook with a lattice and a variable-length code in one calibration-free pipeline for both the linear layers and the KV cache of large language and diffusion transformers. A per-tile Walsh–Hadamard rotation whitens each tile, an optimal low-dimensional lattice (E_8, D_4, A_2, or $\mathbb{Z}$) quantizes it, a lossless strip discards the bits lattice membership already fixes, and a Rice code carries the remainder to within ~ 0.1 bps of that lattice's rate–distortion ideal. Decoding is algebraic, so dimension costs no codebook memory. For the KV cache we add subtractive dither, which under an idealized sampler leaves every cached inner product unbiased per vector, and measurably so in practice, where rotation alone is unbiased only on average.

Table 1 collects operating points. On Llama-3.1-8B weights the default E_8 leads HIGGS at 3–5 bps, by 0.93 ± 0.07 PPL at b=3 over 4 seeds. Neither OCTOPUS nor TurboQuant released usable code, so we reproduce both, validate each against published numbers, and run all three codecs in one harness under one itemized rate accounting over 8 seeds. HYPERQUANT leads both at equal or lower effective rate, apart from a tie with both at 4 bits and a loss to TurboQuant at 2 bits on C4 under QJL; the leads hold at $p \leq 0.016$ over shared seeds. The same pipeline quantizes the 19B-parameter LTX-2 video DiT with no visible artifacts. On one H100 at batch 1 it cuts resident weight memory $\sim 2.8\times$ and the KV-cache encoding rate $\sim 3.8\times$ (computed rather than measured on a filled cache), though the decode kernel stays slower than bare bf16 GEMV.

Table 1: Typical HYPERQUANT operating points; every perplexity row is WikiText-2. Weights and weights+KV rows are Llama-3.1-8B-Instruct against HIGGS's 2-D codebook (dimension-matched in Section 4.4.1); KV rows are Qwen2.5-7B-Instruct-1M against our reproductions, whose two OCTOPUS rows give HYPERQUANT's effective bps then OCTOPUS's. Section 4.4.2 gives the full KV grid, including the 4-bit ties against both baselines and the 2-bit TurboQuant reversal.

Setting	rate	criterion	HYPERQUANT	reference
Weights+KV cache, INT8 MMA	4 bps	PPL ↓	7.50 (+4.8%)	7.16 (bf16)
Weights	4 bps	ΔPPL% ↓	**+3.8%**	+6.4% (HIGGS)
Weights	3 bps	ΔPPL% ↓	**+22.1%**	+33% (HIGGS)
KV cache	2.95/3.13 bps	ΔPPL% ↓	**+4.9%**	+8.1% (OCTOPUS repro.)
KV cache	3.89/4.09 bps	ΔPPL% ↓	**+0.9%**	+1.7% (OCTOPUS repro.)
KV cache	3.8 eff. bps	ΔPPL% ↓	**+1.6%**	+3.3% (TurboQuant repro.)
LTX-2 video	4 bps	LPIPS ↓	0.20–0.21	0 (bf16)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Frontier large-language and generative models (AI, 2024; Lightricks, 2024) routinely exceed tens of billions of parameters and produce KV caches that dominate inference memory at modern context lengths. Both effects make autoregressive decoding memory-bandwidth-bound: each generated token streams the entire weight set and the full KV cache from global memory while performing only a thin matrix-vector product, so arithmetic intensity stays far below the hardware's compute-to-bandwidth ratio and the Tensor Cores sit largely idle (Pope et al., 2023). Post-training quantization (PTQ) turns this bandwidth bottleneck into a *rate-distortion* data-compression problem.

A long line of calibration-based work, GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024), SmoothQuant (Xiao et al., 2023), OmniQuant (Shao et al., 2024), has chipped away at the rate, at the cost of a calibration dataset and per-layer optimization. Data-free schemes are simpler and generally preferred for deployment. The state of the art among them, HIGGS (Malinovskii et al., 2025), Hadamard-rotates weights toward Gaussianity and quantizes to multi-dimensional codebooks that are MSE-optimal for that Gaussian, justified by a linearity theorem that reduces global perplexity damage to per-layer ℓ_2 error. But HIGGS's codebooks are finite Lloyd grids with *fixed-rate* indices; information theory predicts this fixed-rate overhead grows with codebook fineness (shaping gain) and that an entropy-coded quantizer with the same MSE always needs fewer bits (Lookabaugh & Gray, 1989). Our measurements (Section 4.1) confirm this gap is real on Llama weights: HIGGS's index entropy is 0.6–$5.9\,\%$ below its fixed bit budget at 3–5 bps. We close this gap the classical way (Conway & Sloane, 1999; Zamir, 2014): combine a *lattice* quantizer with a variable-length coder over its indices.

KV-cache quantization has converged on a different rotation-plus-marginal scheme. TurboQuant (Zandieh et al., 2026) rotates each head's KV vector and Lloyd-quantizes the resulting Beta-marginal coordinates. OCTOPUS (Boss et al., 2026) extends the marginal trick to coordinate *triplets* via an octahedral parameterization. Both are data-oblivious yet pay the same fixed-rate overhead HIGGS does, on a different marginal. On the lattice side, the closest contemporary is NestQuant (Savkin et al., 2025), which uses nested Gosset lattices plus a calibration-style QA-LDLQ correction, raising Llama-3-8B perplexity by $\sim 3.9\%$ at 4 bps weights+KV.

Contributions. We propose HYPERQUANT, a single calibration-free, post-training pipeline applying the rate-distortion-optimal triplet of whitening, lattice quantization, and entropy coding to both weights and KV cache, integrated with Hopper's FP8/INT8 and Blackwell's NVFP4/MXFP4 MMA paths (Micikevicius et al., 2022; NVIDIA, 2022; 2024), and benchmarked end-to-end on Llama-3.1-8B-Instruct and LTX-2-19B. Our contributions:

- **Per-tile Random Hadamard Transform (RHT)**: input and weight tiles are Hadamard-rotated in blocks matched to the lattice dimension and MMA tile, folded into the preceding norm layer where possible.
- **Lattice + Rice coding**: quantize each rotated tile with E_8 (8-D), D_4 (4-D), A_2 (2-D), or scalar $\mathbb{Z}$, strip the bits lattice membership fixes deterministically (provably within $\lesssim 0.02$ bps of rate-optimal, not exactly rate-optimal – Appendix F.4 gives the residual and its sign-changing origin), and Rice-code the rest at a Gaussian-calibrated rate to within ~ 0.01-bps precision (Section 4.1).
- **Faithful reproductions of both KV baselines**, neither of which released usable code, each validated against a reference outside this paper (Appendices D.14 and D.15). They show OCTOPUS's published table to understate its own method, apparently having been produced with scalar rather than the joint rounding its specification makes default (Appendix D.15).
- A *rate–distortion decomposition* of the HYPERQUANT-vs-HIGGS gap (Section 4.1): at $b{=}4$ a ~ 0.75 dB piece is HIGGS's fixed-rate index redundancy (recoverable by any entropy-coded retrofit) and a smaller ~ 0.36 dB piece is the structural advantage of an unbounded codebook, growing to ~ 1.79 dB at $b{=}5$.
- **A bias-correction menu** for the KV cache (per-layer ± 1-signs or full QJL rotation, optional Schuchman subtractive dither (Schuchman, 1964; Zamir & Feder, 1992)), with a formal proof (Appendix E) that dither is *strictly* inner-product unbiased per cached vector, versus QJL's only distribution-average unbiasedness (Zandieh et al., 2025).
- **A parallel GPU Rice decoder** narrowing the kernel-level gap to bare bf16 GEMV from 6.4–$10.2\times$ to 2.0–$2.2\times$ (Section 3.1 and Appendix D.7).

- A two-regime picture of KV-cache quality (Section 4.2): a high-quality regime (bps ≥ 2.5) where all bias-correction variants are within 0.04 PPL, and a high-compression regime (1.7–2.5 bps) where QJL-style rotation pulls ahead by up to ~ 0.5 PPL; a stress test on the 19B-parameter LTX-2 DiT (Appendix D.6) confirms the pipeline transfers beyond LLMs.

Related work. HYPERQUANT shares HIGGS's rotation+codebook architecture (Malinovskii et al., 2025) but replaces the finite Lloyd codebook with an unbounded lattice addressed by a variable-length Rice code (quantified in Section 4.4.1); it is data-free and orthogonal to calibration-based refinements (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024; Dettmers et al., 2024; Ashkboos et al., 2024a) and to the learned-rotation line (Chee et al., 2023; Tseng et al., 2024; Ashkboos et al., 2024b; Liu et al., 2024a). For the KV cache, QJL, TurboQuant, and OCTOPUS (Zandieh et al., 2025; 2026; Boss et al., 2026) stay strictly scalar (or 3-D) after rotation, where HYPERQUANT's variable-length coding and subtractive dither address the unbounded-codebook and unbiasedness gaps listed above. NestQuant (Savkin et al., 2025) is the closest lattice-based contemporary but needs a calibration-style QA-LDLQ update to reach $+3.9\%$ W+KV cost at matched 4 bps, a 0.7pp edge over HYPERQUANT's data-free $+4.6\%$ (without QA-LDLQ, NestQuant's own ablation is $+7.6\%$). Our LTX-2 stress test (Appendix D.6) is, to our knowledge, the first end-to-end PTQ result on a billion-parameter video DiT (Lightricks, 2024) (extended related work in Appendix A).

2 PRELIMINARIES

This section fixes notation for four classical ingredients; extended background is in Appendix B.

Randomized Hadamard transform (RHT). For $n = 2^k$, the Walsh–Hadamard matrix H_n ($H_1{=}(1)$, $H_{2n} = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} H_n & H_n \\ H_n & -H_n \end{smallmatrix}\right)$) is orthonormal and admits an $O(n \log n)$ butterfly. The randomized version composes it with a random sign diagonal $D = \mathrm{diag}(\pm 1)$: $\mathrm{RHT}_n(x) = H_n D\, x$. Applying $H_n D$ to a deterministic x yields an empirical distribution close to $\mathcal{N}(0, \|x\|^2/n \cdot I)$ with high probability (Ailon & Chazelle, 2009), and because RHT_n is orthogonal, quantizing in the rotated space and inverting after preserves ℓ_2 error while redistributing it from outlier channels into an isotropic shape (Chee et al., 2023; Malinovskii et al., 2025).

Optimal low-dimensional lattices. A lattice $\Lambda \subset \mathbb{R}^n$ is scored by its normalized second moment $G(\Lambda)$ (the per-coordinate MSE of quantizing a uniform point of the Voronoi cell $\mathcal{V}(\Lambda)$, scale-normalized) and its packing density; in every dimension ≤ 8 where the optimum is known, the densest packing also minimizes $G(\Lambda)$: $\mathbb{Z}$ (1-D), hexagonal A_2 (2-D), Schläfli D_4 (4-D), Gosset E_8 (8-D) (Conway & Sloane, 1999); this packing/quantizing coincidence is low-dimensional and need not hold in general. Appendix Table 4 gives their high-rate gap to the Shannon bound: $\mathbb{Z}$ sits 1.53 dB away, and E_8 closes 42% of that gap (0.88 dB remaining) with an $O(1)$ closest-point decoder (Conway & Sloane, 1999, Ch. 20), the highest dimension for which such a decoder is known in closed form.

Rice codes. For a discrete source with entropy $H(X)$, no uniquely decodable code beats $H(X)$ bits/symbol on average (Cover & Thomas, 2006). A fixed-length code over an alphabet of size N pays $\log_2 N$ bits and is undefined for $N = \infty$; a variable-length code has no such limit, which matters because a lattice quantizer's output is an unbounded integer vector. The Rice code (Rice, 1979) is the practical near-optimal variable-length code for non-negative geometric integer sources: given parameter k, a non-negative m is unary-coded $\lfloor m/2^k \rfloor$ followed by $m \bmod 2^k$ in k raw bits, with $k^\star = \lfloor \log_2 \lceil -\ln(2-p)/\ln(1-p) \rceil \rfloor$ optimal for geometric parameter p. Our two-sided, Laplacian-like lattice indices are zig-zag mapped to $\mathbb{Z}_{\geq 0}$ before coding (Section 3), with k calibrated empirically against the realized index distribution rather than from a closed-form p. The resulting code stays within ~ 0.1 bps of the marginal entropy (Appendix F.1), trading that concession for a stateless, table-free $O(1)$ decoder (Appendix F.4).

Bias correction: rotation and subtractive dither. A nearest-neighbor lattice quantizer is deterministic and hence biased, $\hat{x} = x + e(x)$; harmless for weights (absorbed by surrounding affine parameters (Malinovskii et al., 2025)), but attention is *linear* in cached k, v, so a deterministic bias in k systematically shifts attention scores. Two classical fixes: (i) *random rotation* $\hat{x}_{\mathrm{rot}} = S^\top Q_\Lambda(Sx)$,

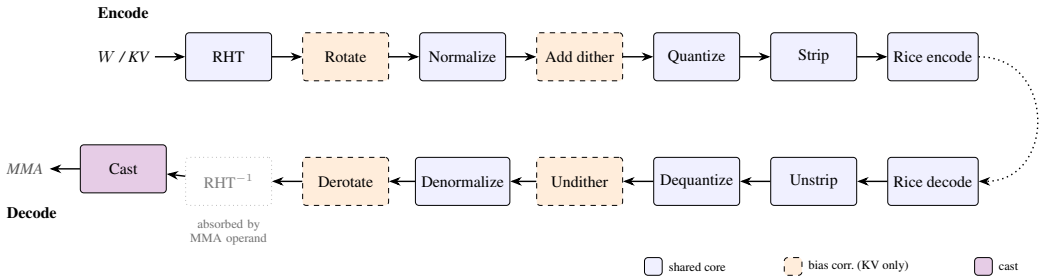


Figure 1: HYPERQUANT end-to-end pipeline. **Encode** (top, left to right) and **decode** (bottom, right to left), each inverse directly below its forward block. Blue is the shared core (weights and KV), orange dashed is bias correction (KV only, ablated in Section 4.2), and purple is the cast to Tensor-Core format. The RHT is absorbed into the matching rotation on the other MMA operand (ghosted); the MMA only consumes the result and is itself outside the codec.

$S \sim \mathrm{Uniform}(\mathrm{O}(n))$ Haar, zero-mean and isotropic *averaged over* S (Zandieh et al., 2025; 2026) but biased for the frozen S actually deployed; (ii) *subtractive dither* (Schuchman, 1964; Zamir & Feder, 1992), $\hat{x}_{\mathrm{dith}} = Q_\Lambda(x+U) - U$ with fresh $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$ per call, exactly uniform on $-\mathcal{V}(\Lambda)$ and independent of x by the Crypto Lemma (Appendix E), giving strict per-vector unbiasedness $\mathbb{E}_U[\langle q,\, \hat{x}_{\mathrm{dith}}\rangle \mid x] = \langle q,\, x\rangle$ for an idealized U; our deployed cube-projection sampler is only $\sim$5% off in second moment (Appendix E.6). The two compose cleanly (Proposition 1): rotation is deterministic-given-x and dither is independent of x.

3 THE HYPERQUANT DESIGN

Figure 1 is the end-to-end block diagram. The *encode* path (top) turns a BF16 tile into a compact code; *decode* (bottom) inverts every active block in reverse and feeds the low-precision MMA. A single encode path serves linear weights (offline, once) and the KV cache (online, per step); they differ only in the two bias-correction blocks (Rotate, Add dither), which run for KV alone.

RHT. Partition x into tiles of size $n_{\mathrm{tile}}{=}2^k$ matched to the MMA unit (128 on H100/Blackwell) and apply (1); the butterfly folds into the preceding norm layer. *Inverse:* none, the rotation cancels against the matching rotation on the other MMA operand ($W = (WH^\top)H$), ghosted in Figure 1. Shared by weights and KV.

$$\mathrm{RHT}_n(x) = H_n D\, x, \qquad D = \mathrm{diag}(\pm 1) \text{ iid uniform.} \tag{1}$$

Rotate (KV only). Optionally rotate by `none`, `signs` ($S{=}\,\mathrm{diag}(\pm1)$, one bit/channel, self-inverse), or `qjl` (Haar S); the best choice tracks the bit-rate (Section 4.2). *Derotate:* apply $S^\top$.

Normalize. Rescale to the lattice's calibration radius, $\tilde{x} = \alpha\sqrt{n}\, Sx/\|Sx\|$ for KV (each (head,token) vector by its own norm), with $\alpha = \alpha(\mathrm{SNR}, \Lambda)$ the closed-form scale realizing the target SNR (Appendix F); being deterministic in x this preserves unbiasedness. *Denormalize:* multiply back by α^{-1} and the stored norm.

Add dither (KV only). Optionally add fresh $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$ before quantizing. *Undither:* subtract the same U after dequantizing, giving strict inner-product unbiasedness (Appendix E).

Quantize. Map $\tilde{x}$ to its nearest lattice point, $O(1)$ for E_8, D_4, A_2 (Conway & Sloane, 1999, Ch. 20), nearest integer for $\mathbb{Z}$. We use four integer realizations $\{E_8^{\mathrm{int}}, D_4^{\mathrm{int}}, A_2^{\mathrm{int}}, \mathbb{Z}_1^{\mathrm{int}}\}$ (Appendix Table 5): $E_8^{\mathrm{int}} = 2E_8 \subset \mathbb{Z}^8$, $D_4^{\mathrm{int}} = D_4 \subset \mathbb{Z}^4$, $A_2^{\mathrm{int}} = \{(\sqrt{3}n_y, n_x) : n_y{+}n_x \equiv 0 \,(\mathrm{mod}\ 2)\}$, $\mathbb{Z}_1^{\mathrm{int}} = \mathbb{Z}$. After α-scaling the coordinates are approximately $\mathcal{N}(0, \alpha^2)$; even at the top of our sweep (5 bps) an 8-bit overflow ($|c_i|{>}127$) is a $\geq 7.3\sigma$ tail event, expecting $\lesssim 10^{-3}$ saturating coordinates across the whole model (Appendix F.2), so the raw code fits one signed byte per scalar.

Strip. Each lattice's membership constraint (parity, coset, or sum modulo a power of two) pins a fixed number of bits per code vector deterministically; Appendix Table 5 gives the resulting lossless savings, up to 1.0 bit/scalar for E_8^{int}. Concretely, for E_8^{int}: coset bit $c{=}c_0 \bmod 2$, halved symbols $s_i{=}(c_i{-}c)/2$, parity $p{=}(\sum_{i\le 6} s_i) \bmod 2$, and $t{=}(s_7{-}p)/2$; the stream $(c, s_0, \ldots, s_6, t)$ recovers all 8 coordinates losslessly. D_4^{int} and A_2^{int} use the analogous even-sum constraint (full constructions in Appendix C). Stripped symbols are zig-zag mapped ($\text{zigzag}(n){=}2n$ for $n{\ge}0$, else $-2n{-}1$) into $\mathbb{Z}_{\ge 0}$ for Rice coding. Removing the strip would cost exactly the stripped bits back (Section 4.1); we show in Appendix F.4 that stripping is provably rate-*optimal*: the marginal entropy of the stripped symbols already meets the lattice ideal $R_D + \frac{1}{2}\log_2(2\pi e\, G(\Lambda))$ to within $\sim 10^{-2}$ bps, so a memoryless coder leaves no exploitable inter-symbol redundancy. *Unstrip:* reconstruct the dropped coordinate from the parity relation and undo the halving.

Rice encode/decode. Entropy-code the stripped symbols at a Gaussian-calibrated rate (Appendix F.1), landing within ~ 0.01 bps of any target. *Cast* (decode only): on the 8-/4-bit MMA path the reconstruction is cast to FP8-E4M3/INT8 (Hopper) or NVFP4/MXFP4 (Blackwell) at the matmul boundary; pure-BF16 deployments skip it.

Parameters. Most knobs have broad sweet spots (Hadamard tile 128–1024, Rice $k \in \{0, 1, 2\}$); practitioners typically only tune target bps and, at low KV bit-rates, the rotation kind. The complete parameter list with defaults and rationale is Appendix Table 6 in Appendix C.

3.1 Implementation

Weights. At load time, each linear layer's weight is RHT-tiled along the input dimension, optionally cast to FP8/INT8, lattice-quantized, and Rice-encoded; codes and per-tile scales are stored. At inference the codes decode on the fly into the MMA input format just in time for the matmul. We do not quantize `lm_head`: it is a small compute fraction and feeds directly into the softmax.

KV cache. Pre-forward hooks on `k_proj`/`v_proj` apply the encode path on the last axis of the reshaped projection output and store the resulting bitstream (our pseudo-quantization harness keeps an equivalent dequantized BF16 tensor for measurement). We hook *pre-RoPE*, since RoPE commutes with per-vector ℓ_2 normalization, and the same argument covers GQA/MQA (AI, 2024). A Haar S costs n^2 floats/layer (64 KiB at $n{=}128$, ~ 4 MiB across Llama-3.1-8B's 32 layers); `signs` costs only n bits and tracks QJL closely at ≥ 2 bps, so we default to `qjl` but offer `signs` as a near-free alternative.

Decoding a variable-length code on the GPU. The Rice code's rate gain comes at a cost: symbol j's bit position depends on every preceding symbol, so a naive decoder is a single serial scan. We adopt an **offset-indexed** decoder: the encoder groups symbols into sub-streams of $S{=}512$ with a per-sub-stream start offset and Rice parameter, decoded independently with the inverse strip and MMA cast fused inline (alternative designs are discussed in Appendix C). The within-sub-stream quotient scan is itself serial, so we further parallelize it with a second decoder we built for this purpose: each shard's carry is an affine function of its carry-in, composed by an $O(\log S)$ parallel prefix scan, with an `__ffs`/`__popc` bit trick replacing the 32-iteration bit-by-bit scan. This reaches **342** Gsym/s on gate-projection, a $3.0\times$ speedup that leaves the kernel within $2.2\times$ of bare bf16 GEMV (down from $6.4\times$), raising HBM utilisation from 5% to 12% (Appendix D.7).

Reference implementation and calibration. The reference implementation is $\sim$3k lines of Python plus the CUDA decoder; a single post-training pass installs the KV hooks and quantizes weights in-place, finishing in ~ 30 s for Llama-3.1-8B on one H100. Since the Rice-coded rate has no closed form, we pre-build a calibration table mapping SNR to empirical Gaussian bps on iid-Gaussian tiles and invert it at inference to hit any target bps to within ~ 0.01, with the realizing per-vector scale α then closed-form in $G(\Lambda)$ (Appendix F). Both are built once, on synthetic data, and transfer unchanged to every tensor since the per-tile RHT plus ℓ_2 normalization renders every tile approximately isotropic Gaussian on a fixed-radius sphere.

4 EXPERIMENTS

We first characterize HYPERQUANT on its own (Sections 4.1 and 4.2, with 8-/4-bit MMA and the LTX-2 stress test in Appendix D.6), then compare head to head against HIGGS (Section 4.4.1) and TurboQuant/OCTOPUS (Section 4.4.2). Quality numbers are exact pseudo-quantization PPL/quality measurements. **Setup:** Llama-3.1-8B-Instruct (AI, 2024) on WikiText-2 raw test (Merity et al., 2017) (141 non-overlapping 2048-token windows, bf16 baseline PPL 7.1606); the OCTOPUS comparison additionally uses Qwen2.5-7B-Instruct-1M at 4096-token context, scored on both WikiText-2 and C4-en (Section 4.4.2); the video experiment uses LTX-2-19B (Lightricks, 2024). All experiments are post-training with no fine-tuning or calibration data beyond the one-time synthetic SNR↔bps table (Appendix F). All runs use one H100 80GB (CUDA 12.4, PyTorch $\geq$2.4; throughput at batch size 1, Appendix D.7).

4.1 WEIGHT-ONLY QUANTIZATION

HYPERQUANT quantizes every `nn.Linear` weight except `lm_head` with per-tile RHT, α-scaling, lattice quantization, stripping, and Rice coding; no FP8 cast, no calibration data. We sweep target rates 3.0–5.0 bps for all four lattices (PPL-vs-rate and SNR-vs-rate curves in Appendix D.1, Appendix Figure 3).

The per-lattice weight SNR (weighted over 224 quantized layers) is constant to ± 0.05 dB across layers and orders exactly as the textbook $G(\Lambda)$: $E_8 > D_4 > A_2 > \mathbb{Z}$ (Appendix D.1), with the best-worst gap shrinking from 0.81 dB at b=3 to 0.65 dB at b=5. We pick E_8 below 3.25 bps, either E_8/D_4 in $[3.5, 4.0]$, and any lattice above 4.25 bps, where choice is below the PPL noise floor. Across all lattices and rates, PPL is a single monotone function of weight SNR, as HIGGS's linearity theorem (Malinovskii et al., 2025) predicts; Section 4.4.1 shows this collapse holds across schemes too.

Does the Gaussian calibration transfer to real weights? Unlike KV vectors (provably Beta-distributed post-rotation), weights have no closed-form guarantee – outlier channels are a real concern here (SmoothQuant, AWQ). Testing directly on 872M actual post-RHT Llama-3.1-8B scalars: empirical kurtosis 3.019 vs. Gaussian's exact 3.0, and ≤ 0.0005 dB SNR cost from the Gaussian calibration at every tested rate (Appendix D.3).

4.2 KV-CACHE-ONLY QUANTIZATION AND BIAS CORRECTION

We benchmark the KV path with bf16 weights, sweeping target bps 1.5–4.0 (rate–quality curve in Appendix D.1, Appendix Figure 4). Our calibration assumes an isotropic Gaussian marginal, though after a Haar rotation the true per-coordinate marginal is Beta-distributed; testing directly with exact Beta-marginal samples shows this costs ≤ 0.01 dB SNR across the high-compression regime, at both n=128 and a smaller head dimension (n=64, where the mismatch would be expected to bite hardest if it mattered at all, Appendix D.2) – no measurable calibration cost from the Gaussian assumption at either head dimension.

The rate sweep splits into two regimes, both measured on Llama-3.1-8B with no residual window: in the high-quality regime ($b \geq 2.5$, ΔPPL 0.05–0.79) every variant is within 0.04 PPL, so the cheapest (`none`) suffices; in the high-compression regime (1.7–2.5 bps, ΔPPL 2.4–13.9) `qjl`/`signs` pulls ahead by ~ 0.5 PPL at b=2.0. HYPERQUANT achieves ΔPPL=+0.25 at b=3.0 with 81% KV-memory reduction, to our knowledge the best published KV-only quality at 3 bps on Llama-3.1-8B without calibration. Below $b \approx 1.8$ each 0.05-bps step roughly doubles the added PPL, setting a practical $\sim$1.7-bps floor for data-free KV quantization.

Bias-correction variants. Appendix Table 7 sweeps all six (rotation $\times$ dither) combinations at E_8, b=2.0 (bf16 weights), the regime where the choice moves PPL (at 4 bps all six are within 0.014 PPL). `qjl` (no dither) is the PPL winner; dither independently improves on plain `none` and buys near-exact per-vector unbiasedness (≈ 0 measured bias, up to the sampler approximation noted in Section 2), while `signs` tracks QJL at $1/128$ the stored memory. We default to `qjl` (or `signs` when storage is a concern), adding dither only when provable unbiasedness is required. That guidance is measured without a residual window; once an exact window is present, as in the Qwen protocol of Section 4.4.2,

Table 2: Llama-3.1-8B end-to-end HYPERQUANT at b=4 with 8-bit MMA, weights-only against weights+KV (`none` bias correction shown); resident-memory figures for this setting are in Appendix Table 13.

MMA cast	Path	PPL	ΔPPL
BF16	—	7.161	—
FP8	weights-only	7.535	+0.37
FP8	weights+KV	7.644	+0.48
INT8	weights-only	7.433	+0.27
INT8	weights+KV	**7.503**	+0.34

the ordering reverses at 2 bits and `none` becomes the better choice (Table 3), plausibly because the window already supplies the recent-token fidelity the rotation was compensating for.

4.3 8-/4-BIT MMA, LTX-2, AND END-TO-END THROUGHPUT

Composing all four components at b=4 with a per-tile FP8 or INT8 cast, adding the KV pipeline on top of weights+8-bit costs only +0.11 PPL (FP8) or +0.07 PPL (INT8) over weights-only (Table 2), and INT8 beats FP8-E4M3 by $\sim$0.10 PPL on this deterministic run, a genuine, consistently-signed effect rather than noise, though smaller across seeds (0.023 ± 0.006 PPL, $4/4$ seeds, Appendix D.10): post-RHT, post-lattice tensors have light (near-bounded) tails, so INT8's uniformly spaced levels are more useful than FP8's logarithmic spacing, an order that flips on raw, outlier-heavy activations. On Blackwell's 4-bit path, NVFP4 (weights+KV, dither) reaches 9.24 PPL (+2.08) at base b=4; MXFP4 needs dither to survive the KV cache at all (Appendix D.5 gives the full NVFP4/MXFP4 table). On the 19B-parameter LTX-2 video DiT (Appendix D.6), the same b=4 weight-only pipeline reaches PSNR 22.74 dB / LPIPS 0.2008 with INT8 MMA (again beating FP8 on every metric) and 3.7× weight-memory reduction with no artifacts visible in our inspection (we did not run a perceptual study or FVD), confirming the pipeline transfers across architectures. End-to-end on one H100, HYPERQUANT compresses the linear weights 3.9× (full-model resident 2.8×) and the KV cache 3.79× at this quality, but this is not yet a throughput win: with the single-thread-per-substream decoder this costs a 3.1× decode-speed regression in one clean, single-process re-measurement (bf16 39.5 → 12.7 tok/s, weights-only; absolute tok/s vary run-to-run on this shared GPU, so we report same-run ratios); the parallel decoder above narrows this to 1.85–1.92× end-to-end, a full-autoregressive measurement consistent with the 2.0–2.2× kernel-level gap and one that already includes the activation-side RHT cost rather than excluding it: every timed decode step calls the rotation unconditionally on the real activation before the GEMV/GEMM, with no caching path that could exempt it (Appendix D.8 traces this in the code). Unfused it is the largest item left: a naive upper bound of ≈13 ms/token summed over all 224 call sites, rivalling the entire bf16 decode budget.

4.4 COMPARISON TO PRIOR WORK

4.4.1 WEIGHTS: HYPERQUANT VS. HIGGS

We compare against HIGGS (Malinovskii et al., 2025) at matched bit rates 3–5 bps: HIGGS runs at native fixed rates with a Lloyd codebook of size $\sqrt{2} \cdot 2^b$ (Pagès & Printems, 2003), HYPERQUANT runs continuously via Rice (Figure 2).

Is the b=3 gap noise? This pipeline's Hadamard transform is deterministic, so the curve above is exactly reproducible; the $\sim$1.3-PPL spread in Appendix D.9 comes from a randomized-RHT variant used only in that tile-size ablation, not the noise floor here (Appendix D.10). Testing the deployed path's one genuinely random mechanism (a per-column sign flip before the Hadamard), paired at 4 seeds, gives 0.93 ± 0.07 PPL at b=3 (13σ) and 0.13 ± 0.07 at b=4, both $4/4$ seeds positive.

At matched dimension (HYPERQUANT's 2-D A_2 against 2-D HIGGS-p2), A_2 still wins at every rate (PPL 9.273 vs. 9.527 at b=3, 7.500 vs. 7.618 at b=4). Converting rate slack to SNR at the 6.02 dB/bps Gaussian slope splits that gap into an *index-entropy* piece, where HIGGS pays $\log_2 N$ bits per index despite lower index entropy and Rice recovers the difference ($0.61 \to 0.18$ dB from b=3 to 5), and an *unbounded-codebook* piece, where a finite codebook must stretch its outermost

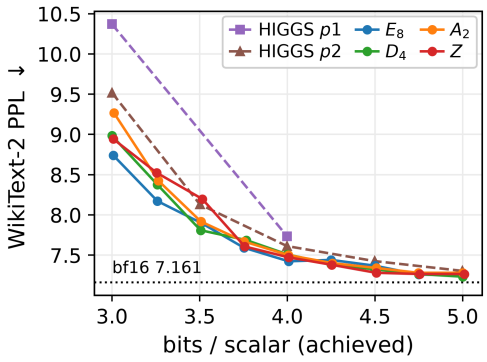


Figure 2: Llama-3.1-8B WikiText-2 PPL vs. bps for HYPERQUANT ($E_8, D_4, A_2, \mathbb{Z}$) and HIGGS ($p \in \{1, 2\}$). E_8, the default, outperforms HIGGS at every bps; scalar $\mathbb{Z}$ trails HIGGS-$p2$ at b=3.5. The four lattices cluster at $b \geq 4.25$ since their asymptotic $G(\Lambda)$ values are within 0.5 dB.

cells over Gaussian tails while the lattice has no boundary cell, so the residual *grows* with rate (0.13 → 1.79 dB) and is realizable only under variable-length coding (Appendix Table 18). Since HYPERQUANT decodes algebraically rather than by table lookup, it also pays no memory penalty for higher-dimensional D_4, E_8 where HIGGS is capped at $p \leq 2$ by shared-memory codebook size, and E_8's lower $G(E_8)$ widens the advantage to 1.3–2.5 dB (8.744 PPL at b=3, 7.434 at b=4).

Two separable advantages. That 1.3–2.5 dB mixes two effects, which Appendix D.4 separates by building fixed-rate Lloyd codebooks at $p \in \{2, 4\}$ and comparing each against its dimension-matched lattice. At matched b=3, entropy coding alone is worth 0.77 dB at p=2 and only 0.28 dB at p=4; the rest comes from E_8's higher dimension, which algebraic closest-point decoding makes free for HYPERQUANT and a shared-memory codebook denies HIGGS. The entropy-coding component is this paper's conceptual contribution and survives at matched dimension; the dimension component follows from HIGGS's cap and would close against a same-dimension fixed-rate codebook that no implementation currently provides.

Component ablation. Isolating each component (Appendices D.9 and D.11), lattice choice follows $E_8 < D_4 < A_2 < \mathbb{Z}$, worth up to 0.57 PPL below b=3.5 and negligible above b=4.25, while RHT tile size has no measurable effect.

4.4.2 KV CACHE: HYPERQUANT VS. IN-HARNESS REPRODUCTIONS OF TURBOQUANT AND OCTOPUS

OCTOPUS (Boss et al., 2026) reports a KV-codec comparison (vs. TurboQuant (Zandieh et al., 2026) and PolarQuant) on Qwen2.5-7B-Instruct-1M at context 4096. Neither baseline's code was available to us, so we reimplemented both from their published specifications and ran all three codecs in one process on that model, sharing the context, rotation draw, seeds, OCTOPUS's two stabilizers (boundary K blocks in BF16, and a 32-token exact residual window costing ≈0.1 bps), and one itemized accounting charging every codec for its code bits, one BF16 per-vector norm, and its amortized rotation (Appendices D.14 and D.15 validate both reproductions against references outside this paper). Quoted p-values are exact paired sign-flip tests over the 8 shared seeds; where HYPERQUANT runs rotation-free it is deterministic, so those tests are one-sample rather than two-arm and reflect the baseline's seed noise alone.

A nominal bit width is not a rate. OCTOPUS's $(b{+}1, b{-}1)$ split spends $(3b{+}1)\lceil n/3 \rceil$ bits per n coordinates, which at n=128 can express 1.34, 2.35, 3.36, or 4.37 bps and nothing between. The blocks of Table 3 are therefore not iso-rate, hence the re-targeted rows, and HYPERQUANT's most aggressive point (2.25 effective bps) has no OCTOPUS counterpart at all (Appendix D.13).

What the comparison shows. Against OCTOPUS at its own realized rates, rotation-free HYPERQUANT leads at b=2 and b=3 on both corpora by 0.053–0.28 PPL ($p \leq 0.016$) while spending

Table 3: HYPERQUANT KV-only vs. our in-harness reproductions of OCTOPUS and TurboQuant, Qwen2.5-7B-Instruct-1M (4096 ctx, $K{=}V$), OCTOPUS's 32-token window, 8 shared seeds, one rate accounting. bps is effective bits per scalar; W2/C4 are ΔPPL% vs. bf16 (6.071 and 11.386), lower better, bold best per column. The *rate-matched* row re-targets the Rice coder to each block's realized OCTOPUS rate, and both HYPERQUANT rotations appear at every rate, so each comparison in the text can be read at matched rotation. Per-seed spreads, rate-matched `qjl`, KV compression, and the windowless ablation: Appendix D.13.

		4 nom. bits			3 nom. bits			2 nom. bits		
codec	rot.	bps	W2	C4	bps	W2	C4	bps	W2	C4
TurboQuant repro.	`qjl`	4.71	+0.6	+0.2	3.75	+3.3	+2.4	2.79	+14.9	+11.0
OCTOPUS repro.	`qjl`	5.06	+0.3	+0.2	4.09	+1.7	+1.4	3.13	+8.1	+5.9
HYPERQUANT	`qjl`	4.72	+0.2	+0.1	3.76	+1.6	+1.2	2.80	+15.4	+13.4
HYPERQUANT	`none`	4.53	+0.1	+0.2	3.58	+2.4	+0.9	2.64	+8.1	+8.0
HYPERQUANT, rate-matched	`none`	4.84	**−0.1**	**+0.0**	3.89	**+0.9**	**+0.3**	2.95	**+4.9**	**+3.5**

0.18–0.21 fewer effective bps. With `qjl` on, HYPERQUANT still leads at $b{=}3$ on both corpora and at $b{=}2$ on WikiText-2, and is nominally behind by 0.018 PPL at $b{=}2$ on C4, within noise.

Against TurboQuant, HYPERQUANT leads at $b{=}3$ at matched rotation and within 0.01 effective bps, by 0.10–0.14 PPL ($p{=}0.008$ on both corpora). At $b{=}4$ both comparisons sit at the level of the baseline's own seed noise: 0.015–0.029 against OCTOPUS's 0.013–0.022 spread, and against TurboQuant 0.027 on WikiText-2 ($p{=}0.008$) but only 0.013 on C4, inside its 0.021 spread ($p{=}0.125$); we read $b{=}4$ as a tie against both and let $b{=}3$ and $b{=}2$ carry the claim. At $b{=}2$ the ordering depends on the rotation: HYPERQUANT's better setting (`none`) leads by 0.34–0.41 PPL while spending 0.15 fewer effective bps, but with `qjl` on both sides TurboQuant leads, by 0.03 on WikiText-2 (within noise) and 0.27 on C4 ($p{=}0.008$). The windowless configuration removes the WikiText-2 crossover (+49.6 against +77.2, Appendix Table 23); we did not run C4 windowless, so that result stands as measured.

One further reading cuts against HYPERQUANT: at matched *nominal* $b{=}2$ its 0.48-bps saving over OCTOPUS is free on WikiText-2 (both +8.1%, $p{=}0.99$) but costs 0.24 PPL on C4. Section 4.2's recommendation of `qjl` under high compression is also wrong here, by 0.45–0.62 PPL.

Why these margins are narrower than the published table would support. OCTOPUS's published Δ% behaves like the windowless configuration: with the window removed our OCTOPUS lands within 0.1 percentage points of its published 2-bit figure and within 1.6 at 3 and 4 bits, and our TurboQuant within 2.2 at 3 and 4 bits. The one large residual is TurboQuant at 2 bits, +77.2% against a published +63.0%. The harness differences Appendix D.13 lists scale with Δ%, large here, and our TurboQuant matches the published *synthetic* distortion at this width to within 0.2% (Appendix Table 24), so we read the residual as harness-level rather than a defect in the reproduction. Giving every codec that window is why our margins are several times narrower than those figures.

5 CONCLUSION AND DISCUSSION

Summary of findings, use, and limitations. HYPERQUANT unifies Hadamard whitening, lattice quantization, Rice coding, and subtractive dither, data-free: $\Delta\text{PPL} \leq 0.3$ at $b{=}4$ (Appendix Table 6), dominance over HIGGS at 3–5 bps (Section 4.4.1), and leads over both reproduced KV baselines at equal or lower effective rate, with the exceptions Section 4.4.2 states. Limitations: Appendix G.

Implications for codebook methods. Our matched-dimension margin over HIGGS splits into a piece any fixed-rate method recovers by entropy-coding its own indices, dominant at low rate, and a piece that requires abandoning the finite codebook, dominant at high rate (Section 4.4.1).

Comparing KV codecs. Nominal bit width does not pin down effective rate once side information is counted, and neither baseline released usable code. Future KV comparisons should report effective bps with side information itemized.

AI USE STATEMENT

Generative AI tools (large language model coding and research assistants) were used throughout this project, at every stage from method design to manuscript preparation.

Tasks with required disclosure. We used generative AI to help develop the pipeline's conceptual framework (Hadamard whitening, lattice quantization, Rice coding, and subtractive dither combined into one construction), to design and iterate on the research methodology and experiments, to implement the method and the GPU decoder, to generate the synthetic calibration data, to formulate and help write up the dither unbiasedness proof (Appendix E), and to interpret results including the rate–distortion decomposition of the HYPERQUANT-vs-HIGGS gap. Translation and qualitative/thematic analysis of interview or survey data are not applicable to this work.

Tasks with recommended disclosure. We additionally used generative AI to edit code and experiment scripts, suggest experimental parameters, create and edit figures, search and summarize related literature, draft and restructure the manuscript (including condensing it to the page limit), format references, and propose the title.

We reviewed all AI-assisted contributions against our own understanding of the method and its results, and take full responsibility for all text, claims, code, and other artifacts in this work.

REPRODUCIBILITY STATEMENT

HYPERQUANT is fully specified in Section 3 (Figure 1, per-block equations, lattice/strip details in Appendix C, calibration in Appendix F); the dither unbiasedness claim is proved in Appendix E, and benchmark configuration and the GPU decoder are in Sections 3.1 and 4. Anonymized code is included with the supplementary material.

REFERENCES

Meta AI. The Llama-3 herd of models. Meta AI research publication, 2024. URL https://ai.meta.com/research/publications/the-llama-3-herd-of-models/.

Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

Saleh Ashkboos, Maximilian L. Croci, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *ICLR*, 2024a. URL https://arxiv.org/abs/2401.15024.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *NeurIPS*, 2024b. URL https://arxiv.org/abs/2404.00456.

Mark Boss, Vikram Voleti, Simon Donné, and Shimon Vainer. OCTOPUS: Optimized KV cache for transformers via octahedral parametrization under optimal squared error quantization, 2026. URL https://arxiv.org/abs/2605.21226. arXiv:2605.21226.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2307.13304.

John H. Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer, 3rd edition, 1999.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2nd edition, 2006.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *ICLR*, 2024. URL https://arxiv.org/abs/2306.03078.

Jarek Duda. Asymmetric numeral systems: Entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013.

Uri Erez and Ram Zamir. On the closeness of the random-dither mapping to the information-theoretic optimum for vector quantization. *IEEE Transactions on Information Theory*, 51(10):3617–3631, 2005.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. URL https://arxiv.org/abs/2403.03206. arXiv:2403.03206.

G. David Forney and Lee-Fang Wei. Multidimensional constellations—Part I: Introduction, figures of merit, and generalized cross constellations. *IEEE Journal on Selected Areas in Communications*, 7 (6):877–892, 1989.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023. URL https://arxiv.org/abs/2210.17323.

Herbert Gish and John N. Pierce. Asymptotically efficient quantizing. *IEEE Transactions on Information Theory*, 14(5):676–683, 1968.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2401.18079.

Lightricks. LTX-Video: A real-time video generation model. GitHub repository, 2024. URL https://github.com/Lightricks/LTX-Video.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *MLSys*, 2024. URL https://arxiv.org/abs/2306.00978.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *NeurIPS*, 2024a. URL https://arxiv.org/abs/2405.16406.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *ICML*, 2024b. URL https://arxiv.org/abs/2402.02750.

Thomas D. Lookabaugh and Robert M. Gray. High-resolution quantization theory and the vector quantizer advantage. *IEEE Transactions on Information Theory*, 35(5):1020–1033, 1989.

Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh. Pushing the limits of large language model quantization via the linearity theorem. In *NAACL*, 2025. URL https://arxiv.org/abs/2411.17525. arXiv:2411.17525.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017. URL https://arxiv.org/abs/1609.07843.

Paulius Micikevicius et al. FP8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022. URL https://arxiv.org/abs/2209.05433.

NVIDIA. NVIDIA hopper h100 architecture white paper. NVIDIA white paper, 2022. URL https://resources.nvidia.com/en-us-tensor-core.

NVIDIA. NVIDIA blackwell architecture technical brief. NVIDIA technical brief, 2024. URL https://resources.nvidia.com/en-us-blackwell-architecture.

Open Compute Project. OCP Microscaling Formats (MX) specification v1.0. OCP specification, 2023. URL https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf.

Gilles Pagès and Jacques Printems. Optimal quadratic quantization for numerics: The Gaussian case. *Monte Carlo Methods and Applications*, 9(2):135–165, 2003.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. URL https://arxiv.org/abs/2212.09748.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xu, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *MLSys*, 2023. URL https://arxiv.org/abs/2211.05102.

Robert F. Rice. Some practical universal noiseless coding techniques. *JPL Publication 79-22*, 1979.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Stosic Dusan, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verrilli, Ralph Wittig, Doug Burger, and Eric Chung. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023. URL https://arxiv.org/abs/2310.10537.

Semyon Savkin, Eitan Porat Chen, Or Lou, and Yury Polyanskiy. NestQuant: Nested lattice quantization for matrix products and LLMs. In *ICML*, 2025. URL https://arxiv.org/abs/2502.09720. arXiv:2502.09720.

Leonard Schuchman. Dither signals and their effect on quantization noise. *IEEE Transactions on Communication Technology*, 12(4):162–165, 1964.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *ICLR*, 2024. URL https://arxiv.org/abs/2308.13137.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *ICML*, 2024. URL https://arxiv.org/abs/2402.04396.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

André Weißenberger and Bertil Schmidt. Massively parallel Huffman decoding on GPUs. In *Proceedings of the 47th International Conference on Parallel Processing (ICPP)*, 2018.

Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7): 560–576, 2003.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *ICML*, 2023. URL https://arxiv.org/abs/2211.10438.

Ram Zamir. *Lattice Coding for Signals and Networks*. Cambridge University Press, 2014.

Ram Zamir and Meir Feder. On universal quantization by randomized uniform/lattice quantizers. *IEEE Transactions on Information Theory*, 38(2):428–436, 1992.

Amir Zandieh, Majid Daliri, and Insu Han. QJL: 1-bit quantized JL transform for KV cache quantization with zero overhead. In *AAAI*, volume 39, pp. 25805–25813, 2025. URL https://arxiv.org/abs/2406.03482. arXiv:2406.03482.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online vector quantization with near-optimal distortion rate. In *ICLR*, 2026. URL https://arxiv.org/abs/2504.19874. arXiv:2504.19874.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. URL https://arxiv.org/abs/1801.03924.

A Extended related work

Weight quantization with rotations and finite codebooks. QuIP (Chee et al., 2023) introduced *incoherence processing*, a randomized rotation that whitens the weight matrix before quantization, enabling 2-bit weight quantization with provable guarantees. QuIP# (Tseng et al., 2024) sharpened this with an E_8-lattice codebook and a quasi-incoherent transform. QuaRot (Ashkboos et al., 2024b) and SpinQuant (Liu et al., 2024a) extended the rotation idea to *learned* 4-D rotations applied online to activations, achieving end-to-end 4-bit inference. HIGGS (Malinovskii et al., 2025) crystallized the data-free version: Hadamard rotation plus a multi-dimensional Lloyd codebook trained on a Gaussian prior. HyperQuant shares the rotation+codebook architecture but differs in two structural respects: the codebook is an infinite lattice, so codeword density is set by a continuous SNR knob rather than a discrete codebook size, and the index stream is entropy-coded via Rice coding rather than transmitted at fixed $\log_2 N$ bits per index; both differences are quantified in Section 4.1.

Calibration-based methods. GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024), OmniQuant (Shao et al., 2024), SmoothQuant (Xiao et al., 2023), SpQR (Dettmers et al., 2024), and SliceGPT (Ashkboos et al., 2024a) use calibration data to refine per-channel scales, salvage outlier features, or solve a Hessian-aware weight allocation problem. HyperQuant is calibration-free by design and sits orthogonal to these methods; a future direction is to compose HyperQuant's bit-allocation knob with an LDLQ-style calibration update (Tseng et al., 2024; Savkin et al., 2025).

KV-cache quantization. Static per-channel methods like KIVI (Liu et al., 2024b) and KVQuant (Hooper et al., 2024) reach 2-bit KV using outlier-aware mixed precision. QJL (Zandieh et al., 2025) introduced a 1-bit data-oblivious quantizer based on a Quantized Johnson–Lindenstrauss sketch, with approximate unbiased inner products. TurboQuant (Zandieh et al., 2026) replaced the QJL sketch by an exact random rotation plus Lloyd-Max scalar quantization on the Beta-distributed unit-vector coordinates, reaching 4–7$\times$ KV-cache compression with $\sim$ zero quality loss on Gemma and Mistral. OCTOPUS (Boss et al., 2026) jointly quantizes triplets of rotated coordinates via an octahedral parameterization, extending the rate–distortion curve to extreme ($\leq$ 2-bit) operating points. All three methods stay strictly scalar (or 3-D scalar) after rotation; HyperQuant instead uses true multi-dimensional lattices (E_8 is 8-D) addressed by a variable-length code, jointly delivering a higher granular gain and an unbounded codebook: the codebook piece alone is worth ~ 0.36 dB over even an entropy-coded HIGGS at b=4, growing to ~ 1.79 dB at b=5, on top of a further ~ 0.49 dB from E_8's extra dimensionality (Section 4.1). On the bias side, TurboQuant inherits QJL's distribution-average unbiasedness; the only strictly per-vector-unbiased construction we know in this space is subtractive dither (Schuchman, 1964; Zamir & Feder, 1992; Erez & Zamir, 2005), which we adopt and prove correct on the full pipeline.

Nested lattices. NestQuant (Savkin et al., 2025) is the closest contemporary work in spirit: like HyperQuant it uses the E_8 Gosset lattice as the quantizer, and like HyperQuant it is motivated by information-theoretic optimality (matrix-product rate–distortion bounds). NestQuant's practical performance, however, relies on a calibration-style QA-LDLQ post-processing step that performs a Cholesky-style update on the unquantized weights based on the quantized activation noise. HyperQuant keeps the data-free property and offsets the absence of that calibration step with an entropy code and a richer rotation menu (none/signs/Haar). Because the two works use different base models and eval harnesses, absolute PPL is not comparable; on a baseline-normalized basis HyperQuant's data-free weight+KV path is competitive in the matched W+KV regime: its E_8 path at 4 bps costs $+4.6\%$ on Llama-3.1-8B, within a fraction of a point of NestQuant's $+3.9\%$ W+KV at ~ 4 bps, which is obtained *with* QA-LDLQ calibration. Their own ablation makes the calibration's role explicit: without QA-LDLQ, NestQuant's W+KV path PPL cost rises to $+7.6\%$.

Diffusion and video transformers. Quantization of diffusion transformers (Peebles & Xie, 2023; Esser et al., 2024; Lightricks, 2024) is less explored than LLM quantization, with most published

numbers focused on image rather than video models. The LTX-2 (Lightricks, 2024) stress test in Appendix D is, to our knowledge, the first end-to-end PTQ result on a billion-parameter video DiT, complementing earlier OCTOPUS results (Boss et al., 2026) on the Wan-1.3B DiT.

Numerical formats. FP8 was standardized by NVIDIA's H100 (Micikevicius et al., 2022; NVIDIA, 2022); the smaller FP4 formats (NVFP4 and the OCP MX-FP4) target the Blackwell generation (Rouhani et al., 2023; Open Compute Project, 2023; NVIDIA, 2024). The key practical difference between the two FP4 formats is the scale encoding (FP8-E4M3 in NVFP4 vs. power-of-2 E8M0 in MX-FP4) and the block size (16 vs. 32). HYPERQUANT targets both formats; our experiments show NVFP4 is the only one quality-viable for KV quantization (Appendix D.5).

B EXTENDED PRELIMINARIES

Lattice	n	$G(\Lambda)$	High-rate gap to Shannon	Decoder $O(\cdot)$
$\mathbb{Z}$	1	$1/12 = 0.0833$	$1.53\,\mathrm{dB}$	$O(1)$
A_2	2	0.0802	$1.36\,\mathrm{dB}$	$O(1)$
D_4	4	0.0766	$1.17\,\mathrm{dB}$	$O(1)$
E_8	8	0.0717	$0.88\,\mathrm{dB}$	$O(1)$
∞-D sphere	—	$1/(2\pi e) = 0.0586$	$0\,\mathrm{dB}$	—

Table 4: Classical lattice quantizer constants for the four lattices used in this paper. The high-rate gap to the Shannon bound is $10\log_{10}(G(\Lambda)\cdot 2\pi e)$. The asymptotic limit $1/(2\pi e)$ is the infinite-dimensional sphere bound. These normalized second moments are scale-invariant, so they apply unchanged to the integer realizations E_8^{int}/D_4^{int}/A_2^{int}/$\mathbb{Z}_1^{\mathrm{int}}$ used in our code.

Nearest-neighbor decoding. For A_2, D_4, and E_8, the closest-lattice-point algorithm follows Conway–Sloane (Conway & Sloane, 1999, Ch. 20): round each coordinate to the nearest integer; if the result violates the lattice's parity constraint, move to the nearest point of the complementary coset (for E_8 the half-integer coset $D_8+\frac{1}{2}\mathbf{1}$, for D_4 and A_2 the parity-flipped neighbor), and keep whichever candidate has the smaller residual norm. The total work per scalar is $O(1)$. E_8 is the highest-dimensional lattice for which nearest-neighbor decoding admits such a constant-time closed-form implementation (Conway & Sloane, 1999).

Granular gain. The fundamental advantage of multi-dimensional vector quantization (VQ) over scalar quantization comes from the freedom to choose the shape of the quantization cell. A scalar quantizer's cell is forced to be an interval, the Voronoi cell of $\mathbb{Z}$, so its normalized second moment is fixed at $G(\mathbb{Z}) = 1/12 \approx 0.0833$. In higher dimensions the optimal Voronoi cell becomes progressively rounder, and the value of G descends toward the ball's limit $G_\infty = 1/(2\pi e) \approx 0.0586$. The ratio $10\log_{10}(G(\mathbb{Z})/G(\Lambda))$ is the *granular gain* of Λ over the scalar quantizer; it is the source-coding counterpart of the (channel-coding) shaping gain familiar from constellation design (Forney & Wei, 1989; Zamir, 2014). In dB terms, the entire budget available to a quantizer, $\mathbb{Z}$'s gap to the Shannon bound, is $1.53\,\mathrm{dB}$. The four lattices used in this paper traverse this budget unevenly: A_2 recovers $0.17\,\mathrm{dB}$, D_4 recovers $0.37\,\mathrm{dB}$, and E_8 recovers $0.65\,\mathrm{dB}$, leaving $0.88\,\mathrm{dB}$ between E_8 and the Shannon bound, i.e. E_8 closes only roughly 42% of the gap. The best lattice we know in any dimension, the 24-D Leech lattice, sits at $G \approx 0.0658$, only a further $\sim 0.38\,\mathrm{dB}$ closer to Shannon than E_8 (Conway & Sloane, 1999; Zamir, 2014): E_8 is not where the gap closes, but rather where the gain-per-decoder-complexity curve sharply flattens.

Entropy coding, expanded. For a discrete source X with pmf p, Shannon's source coding theorem states that the average code length per symbol of any uniquely decodable code is bounded below by the entropy $H(X) = -\sum_x p(x)\log_2 p(x)$, achievable to within a fraction of a bit by practical coders such as Huffman or arithmetic coding (Cover & Thomas, 2006). In a lossy pipeline like ours the picture splits into two stages: a quantizer maps a continuous input to a discrete index, trading distortion for the index's bit count, and a lossless entropy coder then represents the index stream near its entropy; the entropy coder neither distorts the source nor alters the quantizer's distortion characteristics.

C EXTENDED METHOD DETAILS

Membership constraints. The following equations characterize lattice membership and form the basis of the bit-stripping transform of Section 3: E_8^{int} has a coset bit $c \in \{0, 1\}$ with all coordinates sharing parity $c_i \equiv c \pmod 2$ and halved coordinates satisfying $\sum_{i=0}^{7}(c_i - c)/2 \equiv 0 \pmod 2$; D_4^{int} requires $\sum_{i=0}^{3} c_i \equiv 0 \pmod 2$; A_2^{int} requires $n_y + n_x \equiv 0 \pmod 2$; $\mathbb{Z}_1^{\text{int}}$ has no constraint. Each modulo-2 constraint pins one bit of the code vector deterministically given the rest.

Table 5: Integer-coordinate realizations of the four HYPERQUANT lattices, referenced from Section 3. "Bits stripped" is the deterministic information the bit-stripping transform removes losslessly before Rice coding.

	E_8^{int}	D_4^{int}	A_2^{int}	$\mathbb{Z}_1^{\text{int}}$
n	8	4	2	1
Membership constraints	coset + sum-mod-4	sum-mod-2	sum-mod-2	none
Bits stripped / scalar	$\mathbf{1.00}$	$\mathbf{0.25}$	$\mathbf{0.50}$	0
Rice parameters	1 (k_s, with c)	2 (k_s, k_t)	2 (k_{t_y}, k_{n_x})	1

The bit-stripping transform, all four lattices. For E_8^{int} (given in Section 3): output stream $(c, s_0, \ldots, s_6, t)$, net saving 8 bits per 8-D vector $= \mathbf{1.0}$ b/sc. For D_4^{int}: set $p = (\sum_{i=0}^{2} c_i) \bmod 2$ and $t = (c_3 - p)/2$ (the even-sum constraint pins the parity of c_3 given c_0, c_1, c_2); output (c_0, c_1, c_2, t), net saving 1 bit per 4-D vector $= \mathbf{0.25}$ b/sc. For A_2^{int}: set $p = n_x \bmod 2$, $t_y = (n_y - p)/2$; output (t_y, n_x), net saving 1 bit per 2-D vector $= \mathbf{0.5}$ b/sc. For $\mathbb{Z}_1^{\text{int}}$: no stripping, Rice-code each scalar directly. Stripped symbols are signed, so each transform's final step applies the zig-zag map before Rice coding. Every transform is bit-for-bit invertible: the decoder reads the output stream, recovers the halved symbol, reconstructs the dropped coordinate (s_7 for E_8^{int}, c_3 for D_4^{int}) using the parity bit computed from the other symbols, rescales by 2, and, for E_8^{int}, adds back the coset bit.

Rice parameters, all four lattices. Halving a symbol narrows its shifted-geometric distribution and lowers its optimal Rice parameter by one, so a symbol coded at k_s has its halved counterpart best coded at $k_t = k_s - 1$ (checked at runtime). E_8^{int} folds the coset bit c into the low bit freed by halving via $\texttt{comb} = 2 \cdot \text{zigzag}(t) + c$, so all eight symbols share a *single* parameter k_s: c rides for free in a bit Rice would emit anyway (without the fold, t would need its own $k_t = k_s - 1$ and c a separate uncompressed bit). D_4^{int} has no coset bit to fold, so the stream uses *two* levels: k_s for c_0, c_1, c_2 and $k_t = k_s - 1$ for t. A_2^{int}'s two coordinates have different spreads (the $\sqrt{3}$-scaled y-axis is wider), so they too use two levels, k_{t_y} for t_y and k_{n_x} for n_x.

Effect on the achievable bit-rate. At a typical operating point of 21 dB SNR (Gaussian high-rate slope ~ 3.7 bps), the four lattices reach empirical Rice bit-rates of 3.74, 3.77, 3.81, and 3.84 bps respectively (for $E_8^{\text{int}}, D_4^{\text{int}}, A_2^{\text{int}}, \mathbb{Z}_1^{\text{int}}$), within 0.10 bps of the high-rate lattice ideal across all four families. Removing the bit-stripping pass would raise the E_8^{int} rate by 1.0 bps and the A_2^{int} rate by 0.5 bps, exactly the deterministic-information overhead Rice coding can never recover without it. This makes stripping rate-*optimal* rather than just a useful trick: the full argument, that the stripped symbols' marginal entropy already equals the lattice ideal, is proved in Appendix F.4.

GPU decoding of variable-length codes: alternatives considered. Parallel decoding of variable-length codes is well studied in image and video codecs. *(i) Bit-serial entropy decode*, as in H.264/AVC CAVLC/CABAC (Wiegand et al., 2003), parallelizes across independent slices and macroblocks but keeps the entropy stage serial within a stream; its throughput comes from many independent units, not from splitting one stream. *(ii) Fixed-chunk multi-pass synchronization*, used by massively-parallel GPU Huffman decoders (Weißenberger & Schmidt, 2018), cuts the bitstream into fixed-size input chunks decoded provisionally in parallel, then runs synchronization passes exploiting the self-synchronizing property of prefix codes to recover codeword boundaries; it needs no per-stream metadata but pays for multiple passes and seam-convergence logic. *(iii) Offset-indexed one-pass*, our choice (Section 3.1), cuts the symbol stream into many short sub-streams and stores an explicit start bit-offset per sub-stream, so every thread seeks to its offset and decodes a fixed number of symbols

in a single pass. *(iv) Replacing the code* with an interleaved asymmetric-numeral-system (rANS) coder (Duda, 2013) admits N-way SIMD decoding without the serial unary scan at a near-identical rate, but changes the codec itself; we leave this for future work (Section 5). We chose (iii) because it is single-pass and branch-light (no seam synchronization), giving the highest decode throughput of the variants we implemented; its metadata is cheap since the offset+k table shrinks as $32/S$ per symbol ($\lesssim 1\%$ of the ~ 4-bps payload at $S{=}512$); and it composes with the structural decomposition and the MMA cast in the same pass. The trade-off is that S couples compression (large S: less metadata) against parallel occupancy (large S: fewer threads); we use $S{=}512$.

Complete parameter list. Appendix Table 6 lists every knob HYPERQUANT exposes, the values we use as defaults, and a one-line justification.

Table 6: HYPERQUANT's parameter list with defaults, benchmarked in Section 4.

Knob	Default	Rationale
Lattice Λ	E_8	Best 8-D granular gain, $O(1)$ decoder, fits MMA tile.
RHT tile n_{tile}	128	Matches H100/Blackwell MMA K-dim; quality is insensitive to it (Appendix D.9).
Target bps b	4.0 (W), 3.0 (KV)	Sweet spot; $\Delta\text{PPL} \leq 0.3$ vs BF16 at LLM scale.
Rotation kind (KV)	`qjl`	Default at $b \geq 2$ bps; switch to `none` at $b \leq 1.6$.
Dither (KV)	off	Enable when near-exact per-vector unbiasedness is required.

D EXTENDED EXPERIMENTAL RESULTS

D.1 WEIGHT-LATTICE AND KV-ONLY RATE CURVES

Appendix Figure 3 plots the weight-quantization PPL-vs-rate and SNR-vs-rate curves underlying Section 4.1, and Appendix Figure 4 plots the KV-only rate–quality sweep underlying Section 4.2.

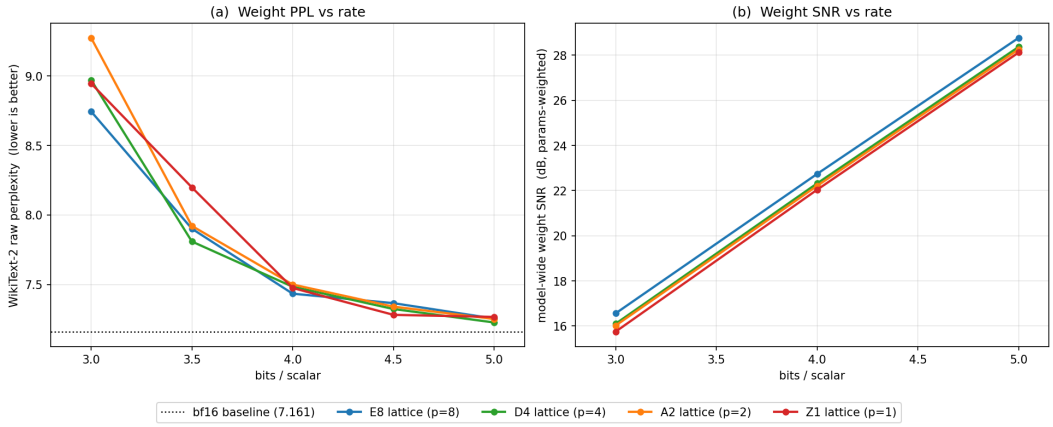


Figure 3: Weight quantization on Llama-3.1-8B at matched bps across the four lattices. **(a)** WikiText-2 PPL vs. rate (bf16 baseline 7.161): at $b \leq 3.25$ the higher-dimensional E_8 wins clearly, while at $b \geq 4.5$ all four cluster within 0.05 PPL. **(b)** Model-wide weight SNR vs. rate, ordering exactly as $E_8 > D_4 > A_2 > \mathbb{Z}$.

D.2 GAUSSIAN CALIBRATION VS. THE TRUE KV MARGINAL

Section 4.2 and Appendix G state the result; here is the measurement. After a Haar rotation, an ℓ_2-normalized KV vector's direction is exactly uniform on the unit sphere, so its per-coordinate marginal, rescaled by $\sqrt{n}$ (n=head dim), is exactly Beta$((n{-}1)/2, (n{-}1)/2)$ on $[-\sqrt{n}, \sqrt{n}]$ – a fact about the rotation, not an empirical property of any particular model, so it can be tested with synthetic

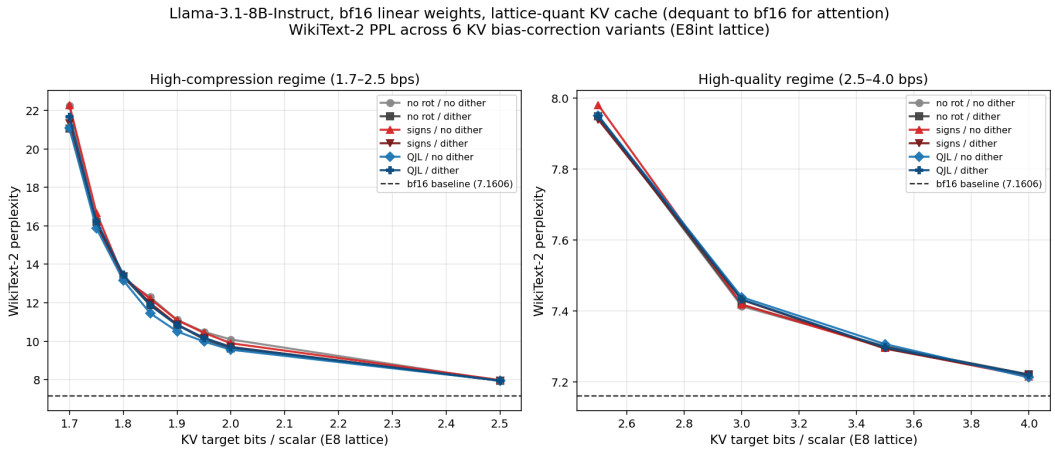


Figure 4: KV-only HYPERQUANT on Llama-3.1-8B, bf16 weights. (Left) High-compression regime (1.7–2.5 bps): QJL/signs rotation pulls ahead of `none`, reaching ~ 0.5 PPL at b=2.0. (Right) High-quality regime (2.5–4.0 bps): all six bias-correction variants collapse onto the bf16 baseline, within 0.04 PPL of one another.

Table 7: Bias-correction variants for HYPERQUANT KV on Llama-3.1-8B (KV-only, E_8 at b=2.0, bf16 weights), the regime where the choice moves PPL. No residual window here, which is why the ordering differs from the Qwen protocol of Section 4.4.2. Summarized in Section 4.2; at 4 bps all six land within 0.014 PPL of one another.

variant	rotation	dither	PPL	ΔPPL vs. bf16	per-vec bias
`none`	–	off	10.089	+2.928	-2.2×10^{-2}
`dither`	–	on	9.622	+2.462	≈ 0
`signs`	±1	off	9.896	+2.735	-4.0×10^{-2}
`signs+dith`	±1	on	9.706	+2.546	≈ 0
`qjl`	Haar	off	**9.557**	**+2.397**	$+1.9\times10^{-2}$
`qjl+dith`	Haar	on	9.685	+2.524	≈ 0

samples and no model at all. For each raw target rate $b \in \{2.0, 2.5, 3.0\}$ (the high-compression regime where a marginal mismatch would bite hardest) we: (i) take the Gaussian-calibrated SNR for E_8^{int} at that rate from the standard calibration table (Appendix F.1); (ii) quantize $50\,000 \times n/8$ i.i.d. E_8^{int}-blocks of true unit-sphere samples (draw a Gaussian direction, normalize, rescale by $\sqrt{n}$) at the resulting α; (iii) compare the achieved SNR against quantizing the same number of i.i.d. $\mathcal{N}(0,1)$ blocks at the identical α. We test both the paper's head dimension (n=128) and a smaller one (n=64, common in some GQA/MQA configurations, where the sphere-marginal-to-Gaussian concentration is weaker and a marginal-aware quantizer would be expected to have its largest advantage if the mismatch mattered at all).

head dim n	target raw bps	SNR on i.i.d. Gaussian	SNR on true Beta marginal	gap
64	2.0	9.744 dB	9.745 dB	−0.001 dB
	2.5	12.848 dB	12.842 dB	+0.006 dB
	3.0	16.575 dB	16.565 dB	+0.010 dB
128	2.0	9.743 dB	9.744 dB	−0.002 dB
	2.5	12.845 dB	12.845 dB	−0.000 dB
	3.0	16.568 dB	16.567 dB	+0.001 dB

Table 8: Gaussian-calibrated α applied to the true post-rotation KV marginal vs. to i.i.d. Gaussian, at $n \in \{64, 128\}$.

Both gaps are at the floor of what this measurement can resolve (800,000 E_8^{int}-blocks per condition) and show no trend with rate. n=64's gaps are slightly larger than n=128's (≤ 0.01 vs. ≤ 0.002 dB)

but still an order of magnitude below anything that would move a PPL number – the high-dimensional concentration that makes a sphere marginal approach Gaussian is already strong enough at both head dimensions that the calibration mismatch TurboQuant's design is motivated by avoiding is not measurable for HYPERQUANT's lattice quantizer even at the smaller, more marginal-sensitive dimension. We do not read this as evidence the mismatch is irrelevant in general (still smaller head dimensions, or codecs more sensitive to tail behavior than a lattice's granular gain, could differ), only that it is not the explanation for any of the gaps observed in Section 4.4.2, at either head dimension tested.

D.3 DOES THE GAUSSIAN CALIBRATION TRANSFER TO REAL WEIGHT TILES?

The test above has a closed-form guarantee to fall back on: a Haar rotation makes the KV marginal *exactly* Beta, so there is no risk the test is confounded by some property of a specific model. Weights carry no such guarantee – outlier channels are a real, data-dependent phenomenon in this literature (SmoothQuant, AWQ, QuIP), and the paper's claim that RHT drives weight tiles toward the Gaussian calibration distribution (Appendix F.3) is a mechanistic argument, not a closed-form identity. We therefore ran the direct, model-dependent version of the test: collected the actual post-`_hadamard_normalize` scalar stream (exactly the quantity Section 3's Quantize step consumes) from a sample of four layers of each kind (`q/k/v/o/gate/up/down_proj`) across Llama-3.1-8B-Instruct – $872{,}415{,}232$ real scalars – and compared quantizing them at the Gaussian-calibrated α against quantizing the same number of i.i.d. $\mathcal{N}(0,1)$ scalars at the same α, at the paper's actual tested weight rates.

target bps	SNR on real weight scalars	SNR on i.i.d. Gaussian	gap
3.0	16.565 dB	16.566 dB	+0.0002 dB
4.0	22.738 dB	22.738 dB	+0.0001 dB
5.0	28.782 dB	28.781 dB	−0.0005 dB

Table 9: Gaussian-calibrated α applied to 872M real post-RHT Llama-3.1-8B weight scalars vs. i.i.d. Gaussian.

The real scalar stream's empirical kurtosis is 3.019 against a Gaussian's exact 3.0, and the SNR gap (≤ 0.0005 dB) is, if anything, tighter than the KV case above – consistent with `hadamard_size=1024` mixing enough real weight coordinates per tile that whatever outlier structure exists in the raw weight is washed out by the rotation, matching the mechanistic argument in Appendix F.3 with a direct measurement on real weights rather than resting on the mechanism alone. This closes the one asymmetry between the paper's KV-marginal rigor and its weight-marginal rigor: the same direct test now exists for both, with the same negative (no measurable mismatch) result.

D.4 HOW MUCH OF THE HIGGS GAP IS ITS DIMENSION CAP?

HIGGS's real deployment caps its Lloyd codebook at $p \leq 2$ for shared-memory reasons – a constraint of HIGGS's specific implementation, not of fixed-rate Lloyd quantization itself. We cannot build an actual higher-dimension HIGGS-style codebook at the paper's tested rates (b=3–5): at p=8, b=3 alone needs 2^{24} codewords, and b=4–5 need 2^{32}–2^{40} – not just impractical to deploy (the point HIGGS's own $p \leq 2$ cap already illustrates) but infeasible to even fit with Lloyd's algorithm or store for evaluation here. Instead we trained actual Lloyd (k-means) codebooks at $p \in \{2, 4, 8\}$ at the largest bit rate tractable for each (2^{16} codewords at p=8, b=1.5, is already the ceiling of what our $O(\text{samples} \times \text{codewords})$ Lloyd loop can fit here) and compared each to its dimension-matched lattice (A_2, D_4, E_8) at the same rate.

p	bps	N (codewords)	Lloyd (fixed-rate) SNR	lattice (entropy-coded) SNR	gap
2	3.0	64	15.24 dB	16.00 dB	+0.77 dB
4	3.0	4096	15.82 dB	16.11 dB	+0.28 dB

Table 10: Fixed-rate Lloyd codebook vs. dimension-matched entropy-coded lattice, at matched bps $= 3$ (the paper's own lowest tested weight rate).

At matched b=3, the gap between an actual fixed-rate Lloyd codebook and its dimension-matched entropy-coded lattice *shrinks* from 0.77 dB at p=2 to 0.28 dB at p=4 – supporting the reading that part of the E8-vs-HIGGS-p2 gap in Section 4.4.1 is HIGGS's dimension cap specifically (closeable in principle by a same-dimension fixed-rate codebook, a hardware/memory-budget fact about HIGGS's implementation) rather than a fixed-rate-vs-entropy-coded gap that is invariant to dimension. We attempted the same comparison at lower bit rates to reach a tractable p=8 point ($N \leq 4096$), but hit a second, independent limitation: the lattice-side Rice-cost calibration itself becomes unreliable at the SNRs these low rates correspond to for $D_4^{\text{int}}/E_8^{\text{int}}$ (the "single-k assumption does not hold" condition in Appendix F.1's calibration sweep fires repeatedly below $\sim$12 dB), which biases the entropy-coded-lattice side of the comparison in a way we do not trust – both p=4, b=2 and p=8, $b \in \{1, 1.5\}$ gave the fixed-rate Lloyd codebook a large, implausible *advantage* over the lattice (-0.7 to -4.9 dB, the wrong sign relative to the rest of this paper's results and relative to A_2's own clean b=3 point), and we do not report these as a dimension-scaling trend. Extending Appendix Table 10 cleanly to p=8 would need either a more careful low-SNR Rice-cost calibration for E_8^{int} or a different tractable-N operating point than the ones we tried; we leave this for future work rather than report a number we do not trust.

D.5 Full-model quantization at 4-bit MMA precision

The Blackwell generation exposes NVFP4 (16-element FP8-scaled blocks) and MXFP4 (32-element power-of-2-scaled blocks). We feed HyperQuant's lattice output into both formats at b=4 and b=3; Appendix Table 11 summarizes the results. Dither has a small effect on NVFP4 (-0.05 PPL, the `none`→`dither` weights+KV rows) but a large positive effect on MXFP4 (-0.33 PPL at b=4); at b=3 the dither rescue on MXFP4 jumps to -5.89 PPL (the model goes from broken to borderline). The pattern is consistent with dither mattering more at higher quantization noise, which both lower bps and the coarser MX-FP4 grid produce.

Format	Path	Bias	PPL	ΔPPL
base b=4				
NVFP4	weights-only	`none`	8.43	+1.27
NVFP4	weights+KV	`none`	9.29	+2.13
NVFP4	weights+KV	`dither`	**9.24**	+2.08
MXFP4	weights-only	`none`	9.46	+2.30
MXFP4	weights+KV	`none`	$\sim$ 18	$\sim$ +11
MXFP4	weights+KV	`dither`	13.61	+6.45
base b=3				
NVFP4	weights+KV	`dither`	**12.81**	+5.65
MXFP4	weights+KV	`dither`	25.42	+18.26

Table 11: Blackwell FP4 path at HyperQuant lattice bases of 4 and 3 bps (ΔPPL vs. BF16 7.161). NVFP4 + dither is the only MXFP4-class configuration that survives the KV cache; MXFP4's E8M0 scale loses too much dynamic range to handle KV tails.

D.6 Beyond LLMs: LTX-2-19B video DiT

To check that the HyperQuant pipeline transfers to a non-LLM transformer architecture we apply it to LTX-2-19B (Lightricks, 2024), a 19B-parameter diffusion transformer (DiT) for text-to-video synthesis with 1370 linear layers totalling 18.87 billion parameters. We quantize all linear weights with HyperQuant at b=4 bps using FP8 or INT8 MMA and otherwise leave the pipeline (Gemma-3-12B text encoder, VAE, scheduler) at BF16. Generation uses 40 diffusion steps, guidance scale 4.0, and the pipeline's default scheduler; for each prompt we fix a single seed and reuse it identically across the bf16 baseline and every quantized configuration, so PSNR/SSIM/LPIPS differences reflect quantization noise rather than sampling variance.

INT8 MMA achieves PSNR 22.74 dB, SSIM 0.8172, and LPIPS 0.2008 at b=4 bps, improving on FP8 on every metric; the INT8-beats-FP8 finding from the LLM experiments (Section 4.3) replicates cleanly on this larger model and different domain. Weight memory shrinks 35.16 $\rightarrow$ 9.5 GiB (3.7×); generation wall-clock is slightly slower (FP8 254.0 s vs. bf16 209.7 s) because the pseudo-

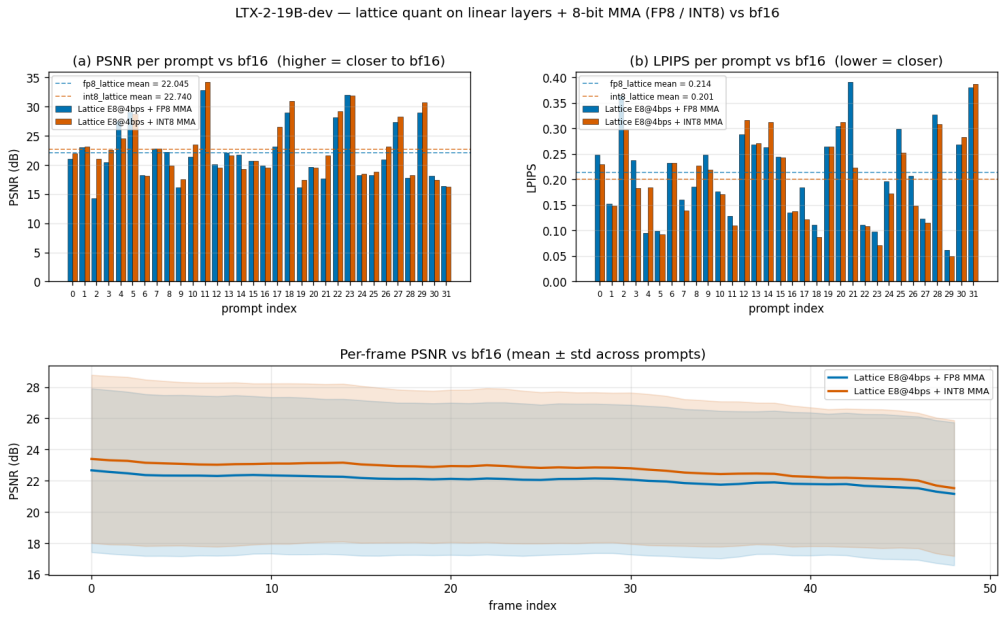


Figure 5: Per-prompt PSNR (a) and LPIPS (b), and per-frame PSNR (mean ± std) for HYPERQUANT on LTX-2-19B versus the bf16 baseline, on a 32-prompt suite at 512×320, 49 frames. INT8 MMA edges FP8 on both PSNR and LPIPS.

Config	PSNR (dB) ↑	SSIM (Wang et al., 2004) ↑	LPIPS (Zhang et al., 2018) ↓
bf16 baseline	(∞)	1.000	0
HYPERQUANT + FP8 MMA	22.04	0.8068	0.2144
HYPERQUANT + INT8 MMA	**22.74**	**0.8172**	**0.2008**

Table 12: LTX-2-19B quality under HYPERQUANT, 32-prompt evaluation. The INT8 MMA path is better than FP8 on every metric at identical b=4 bps.

quantization harness adds a decode pass without exercising MMA acceleration; as on the LLM, HYPERQUANT's win here is the memory reduction rather than wall-clock speed. Appendix Figure 6 shows per-frame PSNR is essentially constant across the 49-frame window: quantization noise does not compound through the DiT's temporal conditioning, and the moderate absolute PSNR (22–23 dB) reflects the natural posterior divergence of a diffusion model under any non-trivial perturbation of its denoising trajectory rather than visible artefacts.

D.7 END-TO-END THROUGHPUT AND MEMORY

Appendix Table 13 reports end-to-end Llama-3.1-8B throughput and resident memory on a single H100 (decode is autoregressive, M=1, timed over 8 warmup + 128 steps; prefill is one 2048-token forward timed over 2 warmup + 5 iterations; batch size 1 throughout), running the single-thread-per-substream decoder of Section 3.1. HYPERQUANT compresses the linear weights 3.9×, cutting full-model resident memory ∼ 2.8× (14.96 → 5.29 GiB); the full-model factor trails the tensor-only 3.9× because the token embeddings and `lm_head` stay BF16 (5.29 GiB is 3.32 GiB of compressed linear weights plus 1.96 GiB of BF16 embeddings/head). The KV-only row's resident memory is essentially unchanged from baseline because the cache is empty at model-load time; its 3.79× savings materialize during generation as the cache fills. Neither path is a throughput win on this hardware: decode is bounded by the serial per-symbol work of the fused decode+GEMV kernel while bf16 cuBLAS is near roofline.

Closing the gap with a parallel decoder. The bottleneck above is that the quotient sub-stream within each of the S=512-symbol groups is still decoded by a single thread as a serial unary scan.



Figure 6: Sample frames from HYPERQUANT on LTX-2: bf16 baseline (top), INT8 + lattice (middle), and per-frame error map (bottom). No visible artefacts; per-frame PSNR is flat across the 49-frame window.

Config	prefill (tok/s)	decode (tok/s)	resident (GiB)	weight cmp.	KV cmp.
BF16 baseline	16,505	51.8	14.96	–	–
HYPERQUANT W (4 bps)	3,915	7.8	**5.29**	3.9×	–
HYPERQUANT KV (4 bps)	16,261	10.8	14.99	–	3.79×
HYPERQUANT W+KV	4,655	5.4	**5.69**	3.9×	3.79×

Table 13: End-to-end Llama-3.1-8B-Instruct on one H100, bf16 base, all at 4 bps, using the single-thread-per-substream decoder of Section 3.1 (i.e., before the parallel decoder optimisation below).

We built a second decoder that removes this residual serialisation: each shard's carry is summarised as an affine function of its carry-in, composed across shards by an $O(\log S)$ parallel prefix scan, and an `__ffs`/`__popc` bit-manipulation inner loop replaces the 32-iteration bit-by-bit scan, so decoding parallelises *within* a sub-stream rather than only across sub-streams. Appendix Table 15 gives its kernel-level benchmark on the same three representative Llama-3.1-8B layer shapes used throughout this paper. Across the three shapes, the parallel decoder reduces the per-layer gap to bare bf16 GEMV from 6.4–10.2× down to 2.0–2.2×, a 3.0–4.8× kernel-level speedup, and raises HBM utilisation from 5% to 12%.

Full autoregressive re-measurement. Absolute tok/s on this shared GPU vary somewhat run to run, so Appendix Table 14 reports one clean, single-process, same-run comparison in which baseline, single-thread, and parallel decoder are measured back to back and only within-run ratios

are treated as meaningful: baseline decode is 39.5 tok/s, the single-thread-per-substream decoder is 12.7 tok/s (3.10× slower than baseline), and the parallel decoder reaches 20.6–21.3 tok/s depending on the prefill-GEMM dtype, i.e. 1.85–1.92× slower than baseline and 1.62–1.67× faster than the single-thread decoder in the same run. This full-autoregressive measurement is in the same ballpark as the 2.0–2.2× kernel-level gap above. We have not measured this at batch size > 1 or alongside FlashAttention-style tiled KV loads; closing the remaining $\sim 1.9\times$ gap (layout changes for coalesced remainder access, a lookup-table byte decode, or – per Appendix D.8 – fusing the activation-side RHT, which this measurement does *not* isolate from the decode kernel itself) remains the clearest concrete path from today's memory win to an outright throughput win.

Config	decode (tok/s)	vs. baseline	vs. serial
BF16 baseline	39.5	1×	–
HYPERQUANT W, single-thread decoder (4 bps)	12.7	3.10× slower	1×
HYPERQUANT W, parallel decoder, fp32 prefill-GEMM	20.8	1.90× slower	1.63× faster
HYPERQUANT W, parallel decoder, bf16 prefill-GEMM	20.6	1.92× slower	1.62× faster
HYPERQUANT W, parallel decoder, int8 prefill-GEMM	21.3	1.85× slower	1.67× faster

Table 14: Single-process, same-run autoregressive decode throughput (weights-only, 4 bps, batch 1); ratios are computed within this one run (same process, same load) to control for run-to-run variation on the shared benchmark machine.

Kernel	q_proj (4096×4096)	gate_proj (14336×4096)	down_proj (4096×14336)
Bare bf16 GEMV	622	735	694
Single-thread-per-substream decoder	61	114	108
Parallel decoder (this work)	**294**	**342**	**340**

Table 15: Fused decode+GEMV throughput (Gsym/s, higher is better) on three representative Llama-3.1-8B layer shapes. The parallel decoder is 3.0–4.8× faster than the single-thread-per-substream kernel of Section 3.1 and within 2.0–2.2× of bare bf16 GEMV, versus 6.4–10.2× before.

D.8 ACTIVATION-SIDE RHT LATENCY

Section 3 states the RHT "folds into the preceding norm layer where possible" without measuring the fold's latency or its coverage. We isolate the routine every `RiceLinear.forward` call actually invokes on its input activation (sign flip, chunked Hadamard via a cached-matrix matmul, rescale) in the released integration, at the two input widths present in Llama-3.1-8B (4096 for q/k/v/o/gate/up-projections, 14336 for down-projection), at batch-1 decode ($T{=}1$) and a 256-token prefill chunk, 200 timed repetitions after 20 warmup calls, CUDA-event timed.

input width	decode ($T{=}1$)	prefill chunk ($T{=}256$)
4096	59.9 μs	71.7 μs
14336	59.7 μs	209.0 μs

Table 16: Activation-side RHT latency (mean of 200 calls) in isolation.

At $T{=}1$ the latency is $\approx$ 60 μs *regardless of input width* – an $\approx 3.5\times$ difference in tensor size (and FLOPs) produces no measurable latency difference, meaning the cost is dominated by fixed dispatch/kernel-launch overhead across the handful of separate PyTorch ops in the routine (multiply by signs, reshape, matmul against the cached Hadamard matrix, reshape, rescale), not by compute. This is a different bottleneck from the one already fixed in this codebase (a Python-loop butterfly of $O(\log n)$ sequential steps, replaced by the single cached-matrix matmul this measurement already reflects); eliminating it further would require fusing the whole routine into one kernel (e.g. `torch.compile` or a custom CUDA op), which we have not done. Naively summed across all 224 `nn.Linear` call sites in one Llama-3.1-8B forward (32 layers × 7 linears), this is ≈ 13.4 ms/token – comparable to or larger than the entire bf16-baseline decode budget ($\approx$ 17–25 ms/token depending on run,

Appendix Table 13). We stress this ≈ 13.4 ms/token figure is a naive upper bound assuming zero fusion coverage (every one of the 224 call sites pays the isolated $\approx 60\,\mu$s cost independently), not a measurement of exactly how many of those sites the "folds into the preceding norm layer where possible" language actually covers in a real forward pass – we did not instrument a per-site fold/no-fold ablation. What we *can* state with certainty, by tracing the code rather than estimating: this cost is not free, and it is not an *additional*, uncounted cost on top of Appendix Table 14's end-to-end decode numbers – it is already inside them. `RiceLinear._forward_bf16` and `_forward_ffs` both call `_rht` unconditionally on the real input activation before their GEMV/GEMM branch, with no conditional that could skip it; the decode-throughput benchmark (Appendix D.7) times the model's actual `forward()` on every decode step, not a stub; and the only caching in `RiceLinear` (`set_weight_cache`) caches the *decoded weight* for the FFS prefill path only, is never invoked by the decode benchmark, and in any case never touches `_rht`, which operates on the per-token activation and cannot be cached across tokens. So the 1.85–$1.92\times$ figure in Section 4.3 already reflects whatever RHT overhead the current, largely-unfused integration pays; this overhead level plausibly explains why the cross-run throughput numbers in Appendix Table 15's predecessor benchmarks were noisier than the kernel-level numbers alone would suggest: a $\sim 60\,\mu$s fixed tax at every one of 224 call sites is present identically regardless of which Rice-decode kernel is substituted in, diluting any kernel-level gain by a constant amount every run.

D.9 Ablation: RHT tile size

The RHT tile is the block length over which we whiten and per-tile `amax`-scale before lattice quantization. We sweep it over $\{128, 256, 512, 1024, 2048\}$ on Llama-3.1-8B with E_8^{int} at $b \in \{3, 4\}$ on the weight, KV, and joint W+KV paths (Appendix Figure 7). To measure the tile effect rather than the alignment of any one fixed Hadamard basis, we use a *true* RHT (a per-tile random sign diagonal applied before the Hadamard) and repeat the sweep over four independent RHT seeds, reporting the across-seed mean ± 1 standard deviation.

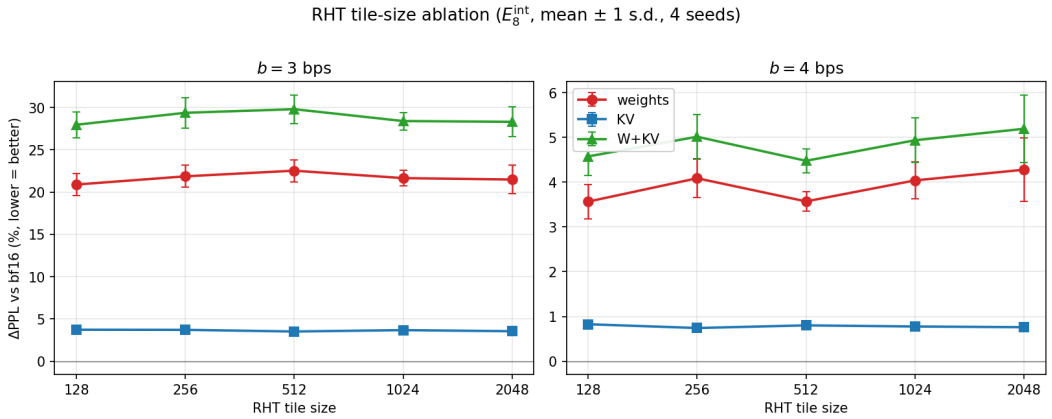


Figure 7: PPL vs. RHT tile size on Llama-3.1-8B (E_8^{int}, no MMA cast, true RHT), as ΔPPL relative to the BF16 baseline for the weight, KV, and joint W+KV paths at b=3 (left) and b=4 (right). Markers are the mean over four RHT seeds; error bars are ± 1 standard deviation.

KV is essentially tile-insensitive: per-tile means span only 3.5–3.7% at b=3 and 0.74–0.83% at b=4, with tiny seed variance (≤ 0.24 pp, mostly < 0.1); KV vectors are low-dimensional and well-conditioned per head, so whitening block size barely matters once the tile covers a head or more. **For weights and W+KV, tile size does not reliably move quality**: across seeds the weight path stays at $\approx$ 21–22.5% at b=3 and $\approx$ 3.6–4.3% at b=4, with per-tile gaps (≤ 1.6 pp) smaller than the ± 1 s.d. seed noise (≈ 1.3 pp at b=3, 0.2–0.7 pp at b=4); apparent "best" tiles from any single basis or seed do not survive averaging over rotations. The W+KV damage tracks the sum of the independent paths (e.g. at b=4, tile 128: weights +3.57% and KV +0.83% compose to W+KV +4.58%), confirming the two error sources are roughly additive in PPL. Because the tile has no reproducible effect on quality, we set it on kernel-efficiency grounds: the default tile of 128 matches the MMA K-dim.

D.10 MULTI-SEED SIGNIFICANCE OF THE WEIGHT-VS-HIGGS GAP

The seed variance quoted above ($\approx$ 1.3 pp at b=3) comes from a *true* RHT variant (a per-tile random sign diagonal, used only for this tile-size ablation), a different mechanism from the b=3 HYPERQUANT-vs-HIGGS gap reported in Section 4.4.1, which uses a fully *deterministic* pipeline (plain Walsh–Hadamard, no random sign draw anywhere in the weight pseudo-quantizer or in HIGGS's stock grid); the two numbers describe different mechanisms and are not directly comparable as a signal-to-noise check. We confirmed the determinism directly: running the Section 4.4.1 pipeline twice on the identical config reproduces the PPL to full floating-point precision (baseline 7.161151885986328 on both runs), i.e. there is no seed to average over in that specific harness.

To test the gap's significance on a mechanism that *does* carry genuine randomness, we add the one such mechanism the deployed weight path has: a per-input-column random ± 1 sign flip D applied before the Hadamard transform (matching `integrations.llama.rice_linear._get_signs`, whose seed is hardcoded in the released integration and not threaded through `convert_linears`), applied identically to both HYPERQUANT and HIGGS at each of 4 seeds, then undone after reconstruction (D is self-inverse). Appendix Table 17 reports the paired result per seed: at b=3 the gap is 0.934 ± 0.072 PPL across seeds (mean$\pm$s.d.), positive in all 4 seeds, a 13.0σ effect by the seed-to-seed standard error; at b=4 it is 0.127 ± 0.066 PPL, smaller and only a 1.9σ effect, though still positive in all 4 seeds. We also multi-seed-checked the INT8-vs-FP8 finding (Section 4.3) the same way, applying the per-tile INT8/FP8 quant-dequant simulation on top of the seeded b=4 E_8 reconstruction: the FP8-minus-INT8 gap is 0.023 ± 0.006 PPL, positive in all 4 seeds, a 3.9σ effect – smaller than the 0.10 PPL figure quoted from the single deterministic run, but a real and consistently-signed effect, not noise.

bps	E_8 PPL (mean$\pm$s.d.)	HIGGS-p2 PPL (mean$\pm$s.d.)	gap (mean$\pm$s.d.)	effect size
3	8.700 ± 0.070	9.634 ± 0.107	0.934 ± 0.072	13.0σ, 4/4 seeds
4	7.451 ± 0.030	7.579 ± 0.038	0.127 ± 0.066	1.9σ, 4/4 seeds

Table 17: Multi-seed (paired, 4 seeds) confirmation of the HYPERQUANT-vs-HIGGS-p2 gap on Llama-3.1-8B, using a per-column random sign flip before the Hadamard transform (the one mechanism in this pipeline with genuine seed-to-seed randomness) in place of the deterministic transform used in Section 4.4.1.

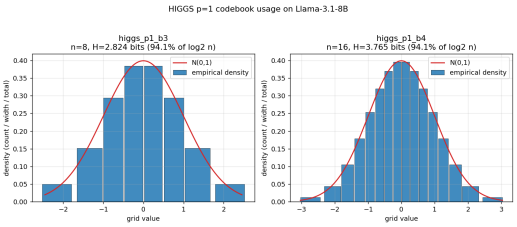
D.11 HIGGS CODEBOOK EFFICIENCY ANALYSIS

To corroborate the rate-gain decomposition of Section 4.4.1, we measured HIGGS's empirical codebook index histograms on Llama-3.1-8B weights (Appendix Figure 8). The Lloyd grids exhibit clear non-uniform usage: the most-used codeword is consistently 10–22$\times$ more frequent than the least-used, and the empirical index entropy is 0.6–5.9% below the fixed-rate budget $\log_2 N$ (equivalently, 94–99% coding efficiency). This is not a bug in HIGGS; it is the expected behavior of any finite codebook on a smooth distribution. It is, however, a recoverable bit-rate gap, which Rice coding closes.

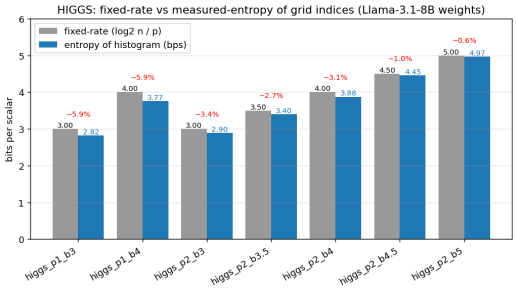
D.12 WEIGHT-VS-HIGGS: SUPPORTING FIGURES

Table 18: Dimension-matched decomposition of the A_2-vs-HIGGS-p2 SNR gap (both 2-D), converted to dB at the local 6.02 dB/bps Gaussian slope. Summarized in Section 4.4.1.

bps	HIGGS-p2 SNR	A_2+Rice SNR	total Δ	entropy-coding piece	unbounded-codebook piece
3.0	15.28 dB	16.02 dB	0.74 dB	0.61 dB	**0.13 dB**
4.0	21.10 dB	22.21 dB	1.11 dB	0.75 dB	**0.36 dB**
5.0	26.29 dB	28.26 dB	1.97 dB	0.18 dB	**1.79 dB**

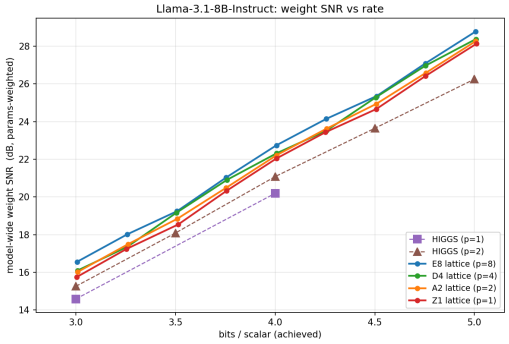


(a) Index frequency for HIGGS p=1, b=3.

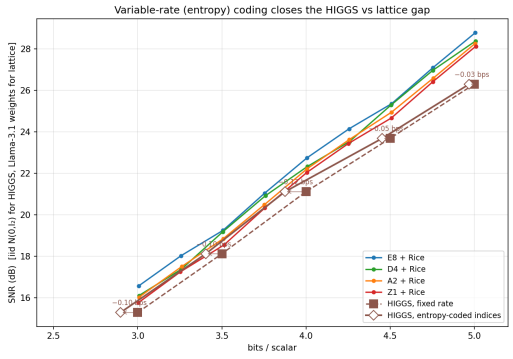


(b) Empirical entropy efficiency H/b for HIGGS codebooks at $b \in \{3, \ldots, 5\}$, $p \in \{1, 2\}$.

Figure 8: HIGGS codebook index distributions are non-uniform on Llama weights, with 0.6–5.9% entropy slack relative to the fixed-rate budget $\log_2 N$. HYPERQUANT recovers this slack via Rice coding.



(a) Weight SNR (dB) vs. bps. HYPERQUANT dominates by 1.3–2.5 dB across the entire range.



(b) Entropy slack: fixed-rate budget minus the empirical index entropy of the HIGGS codebook. The lattice + Rice path closes this slack by entropy coding.

Figure 9: The two components of the HYPERQUANT-vs-HIGGS gap, corresponding to Appendix Table 18.

D.13 KV CACHE: RESIDUAL-WINDOW ABLATION AND TURBOQUANT/OCTOPUS VALIDITY

Appendix Table 19 isolates the effect of the 32-token residual window on HYPERQUANT. The window adds ≈ 0.1 bps and so trims KV$\times$ slightly, but the quality gain grows sharply as the rate falls, exactly where recent-token fidelity matters most, and it is the whole story at the aggressive end: $+42.0\%$ becomes $+7.4\%$ at 2 bps and $+325.1\%$ becomes $+26.9\%$ at 1.7. Since the window is part of OCTOPUS's own protocol, we give it to every codec. Appendix Table 23 runs all three codecs both ways and shows that the window helps each of them, most at the lowest rate, so it is not a stabilizer that happens to suit HYPERQUANT. That table also supplies the strongest external check in this paper: with the window removed, our two reproductions land close to the figures OCTOPUS publishes for both methods.

The 1.7-bps row is the operating point Section 4.4.2 reports without an OCTOPUS comparison. Its effective rate is 2.25 bps once the window, the BF16 per-vector norm, and the boundary-K protection are counted, which is below 2.35 bps, the lowest code rate OCTOPUS's $(b{+}1, b{-}1)$ split can express without giving the triplet norm zero bits. There is therefore no OCTOPUS configuration at this rate to compare against, and we make no comparative claim at this point.

OCTOPUS's published figures, for traceability. We reprint (Boss et al., 2026, Table 2) in Appendix Table 20 because it is the external reference our windowless rows are checked against in Appendix Table 23, and a reader should be able to see that check without leaving this paper. We

bps	$\Delta\%$ (no window)	$\Delta\%$ (window 32)	KV×
4	$+0.8$	$\mathbf{+0.1}$	$3.7 \to 3.6$
3	$+5.5$	$\mathbf{+1.8}$	$4.8 \to 4.6$
2	$+42.0$	$\mathbf{+7.4}$	$6.6 \to 6.4$
1.7	$+325.1$	$\mathbf{+26.9}$	$7.5 \to 7.1$

Table 19: Effect of the 32-token residual window on HYPERQUANT (`none`, WikiText-2 $\Delta\%$).

do not use these numbers for any head-to-head claim, since absolute $\Delta\%$ is not comparable across harnesses that differ in evaluation slice, stabilizers, and V-side codec.

Table 20: OCTOPUS's own published KV comparison (Boss et al., 2026, Table 2) on Qwen2.5-7B-Instruct-1M at context 4096, reprinted as the external reference for the windowless check in Appendix Table 23. All head-to-head claims in this paper come from Table 3 instead.

nom. bits	codec	W2 $\Delta\%$	C4 $\Delta\%$
4	TurboQuant-MSE	$+3.1$	$+1.7$
	OCTOPUS	$+2.7$	$+1.5$
3	TurboQuant-MSE	$+8.6$	$+8.3$
	OCTOPUS	$+7.2$	$+5.9$
2	TurboQuant-MSE	$+63.0$	$+77.4$
	OCTOPUS	$+34.7$	$+41.5$

Threats to validity in the baseline comparisons, and how each is closed. Four gaps threaten any cross-paper KV codec comparison, and all four bear on Table 3's TurboQuant/OCTOPUS rows: rate mismatch (different effective bit-widths at matched nominal bit-width), accounting basis (is side information counted the same way for every method?), non-reproduction (numbers copied from a competitor's own table), and configuration mismatch (a baseline run with OCTOPUS's window bolted on rather than its own native configuration). We close all four for *both* baselines by building in-harness reproductions of each rather than reasoning about the gaps in the abstract. Concretely:

1. *Rate matching.* Nominal bit-width is not a rate, for the reason Section 4.4.2 gives, and neither fixed-rate baseline can hit an arbitrary fractional rate the way HYPERQUANT's Rice coder can. We therefore report each codec's achieved effective bps and additionally re-target HYPERQUANT to each OCTOPUS operating point's realized rate, giving a comparison matched on rate rather than on nominal bit-width.
2. *Accounting basis.* One function (`hyperquant.kv_accounting.itemized_kv_budget`) computes effective bps for all three codecs from the same three components: the quantizer's own code bits, one BF16 per-vector norm (which all three need to undo the pre-quantization ℓ_2 normalization), and the amortized rotation. Each codec reports code bits only, so no codec is charged twice for its norm.
3. *Non-reproduction.* Both baselines are reimplemented rather than quoted, under the shared protocol of Section 4.4.2. Appendices D.14 and D.15 then check each reproduction against references outside this paper: classical high-rate theory for the Lloyd-Max codebook, and OCTOPUS's own published synthetic distortion table for both codecs.
4. *Configuration mismatch.* We run at *both* TurboQuant's native (no window) and OCTOPUS's windowed configuration (Appendix Table 23) rather than only the windowed one.

The residual fidelity gaps are that both reproductions follow published specifications rather than released code. Appendix D.15 states what this does and does not settle for OCTOPUS, including that our codec quantizes V with the same codec as K where OCTOPUS pairs its K codec with a fixed group-of-32 scalar V codec. For TurboQuant the analogous gap is whether its released code allocates rate jointly across coordinates rather than per-coordinate, which we could not verify; what we can say is that our implementation reaches the distortion OCTOPUS independently publishes for TurboQuant-MSE at all three deployed bit widths (Appendix Table 24).

D.14 IS OUR TURBOQUANT REPRODUCTION A CORRECT LLOYD-MAX QUANTIZER?

TurboQuant's own codebase and published rate–distortion curve are not available to us, so we cannot validate `hyperquant.turboquant.beta_marginal_lloyd_codebook` against TurboQuant's own numbers directly. What we can check without that dependency is whether it is a correct, converged Lloyd-Max quantizer for the density it targets – the Beta$((n-1)/2, (n-1)/2)$ marginal, n=128 – via two independent, purely synthetic tests: (i) does the production (quantile-initialized) codebook's distortion track Bennett's classical high-rate asymptotic formula for the *optimal* scalar quantizer of this exact density, $D(N) \sim \left(\int f(x)^{1/3}dx\right)^3/(12N^2)$, computed by independent numerical integration of the density; and (ii) does re-running the identical Lloyd fixed-point iteration from several random initializations, rather than the production quantile init, converge to the same distortion?

N (levels)	production MSE	Bennett prediction	ratio	MSE drop from $N/2$
4	9.067×10^{-4}	1.298×10^{-3}	0.70	–
8	2.653×10^{-4}	3.244×10^{-4}	0.82	3.42
16	7.279×10^{-5}	8.110×10^{-5}	0.90	3.65
32	1.917×10^{-5}	2.028×10^{-5}	0.94	3.80
64	4.918×10^{-6}	5.069×10^{-6}	0.97	3.90

Table 21: Production codebook's MSE vs. Bennett's independent high-rate prediction, n=128. The last column is the factor by which the MSE falls when N doubles, which an optimal scalar quantizer must drive to 4.

Check (i) is clean: the ratio rises monotonically from 0.70 at N=4 toward 1 as N grows ($0.82 \to 0.90 \to 0.94 \to 0.97$), approaching Bennett's asymptote from below and never crossing it. That is the required behavior for a correct high-rate-optimal quantizer, since Bennett's leading-order formula is exact only in the $N \to \infty$ limit, so the finite-N deficit has to shrink monotonically. A ratio that rose *past* 1 at the largest N would indicate the opposite diagnosis, an unconverged Lloyd fit rather than a limitation of the asymptotic formula; we therefore treat convergence to 1 from below as the pass criterion, and iterate each codebook until no codeword moves rather than to a fixed iteration budget. The last column reads the same conclusion off the data without reference to Bennett's constant at all: the MSE drop per doubling of N climbs steadily through $3.42 \to 3.65 \to 3.80 \to 3.90$ toward the factor of 4 that an optimal scalar quantizer must approach.

Check (ii) also passes: re-running the identical Lloyd fixed-point iteration from 5 independent random initializations, under the same codeword-movement stopping rule, reaches the same distortion as the production quantile initialization at every N tested. The largest spread across all six fits (production plus five random) at any single N is 1.5×10^{-8}, four to five orders of magnitude below the MSE itself. For a symmetric, unimodal, bounded density this is the expected outcome, and it rules out the failure mode the check was looking for: a quantile-initialized fit sitting in a favorable local optimum that the reported ratios silently depend on.

Two implementation details turn out to matter for both checks, and we record them because either one alone is enough to make a correct Lloyd-Max implementation look wrong. The initialization must be sorted before the first assignment step, since a 1-D Lloyd assignment by midpoint thresholds is only well defined for an ordered codebook, and an unordered one produces non-monotone thresholds whose misassigned cells look exactly like the classical empty-cluster pathology. The iteration must also stop on codeword movement rather than at a fixed count, since the number of iterations Lloyd needs grows with N: a budget generous enough at N=4 leaves the N=64 codebook about 30% short of its fixed point, which shows up as precisely the above-1 ratio that check (i) is designed to flag. The rates this paper's KV comparison actually deploys are $N \in \{4, 8, 16\}$ (2, 3, and 4 bits per coordinate), and Appendix D.15 closes the loop on those three by checking the same quantizer against a distortion figure a third party publishes for TurboQuant, rather than against asymptotic theory.

Why the window matters for how the margins are read. Setting a windowed HYPERQUANT against OCTOPUS's published $\Delta\%$ would put $+7.4\%$ against $+34.7\%$ at 2 bits, a gap several times the one we report. That comparison crosses protocols, and Appendix Table 23 shows how much

Table 22: Full in-harness grid behind Table 3: absolute PPL, mean±s.d. over the same 8 seeds, on Qwen2.5-7B-Instruct-1M at context 4096 with OCTOPUS's 32-token window and boundary-K protection. bf16 baselines are 6.071 (WikiText-2) and 11.386 (C4). Effective bps is the itemized figure (code bits, one BF16 per-vector norm, amortized rotation). The `none` rows draw neither a dither nor a rotation, so they are deterministic across seeds and their s.d. is exactly zero. *rate-matched* rows re-target HYPERQUANT's Rice coder to the block's realized OCTOPUS rate. The $\Delta\%$ figures elsewhere in the paper are computed from unrounded means, so recomputing one from the three-decimal PPLs printed here can move the last digit by one: the b=3 HYPERQUANT `none` C4 cell of Table 3 is the single case in that table where it does.

nom. bits	codec	rot.	eff. bps	W2 PPL	C4 PPL	KV×
4	TurboQuant repro.	`qjl`	4.706	6.109 ± 0.016	11.411 ± 0.021	3.40
	OCTOPUS repro.	`qjl`	5.058	6.092 ± 0.022	11.404 ± 0.013	3.16
	HYPERQUANT	`qjl`	4.715	6.082 ± 0.011	11.398 ± 0.009	3.39
	HYPERQUANT	`none`	4.530	6.079 ± 0.000	11.411 ± 0.000	3.53
	HYPERQUANT, rate-matched	`qjl`	5.069	6.076 ± 0.011	11.394 ± 0.014	3.16
	HYPERQUANT, rate-matched	`none`	4.844	6.063 ± 0.000	11.389 ± 0.000	3.30
3	TurboQuant repro.	`qjl`	3.750	6.272 ± 0.039	11.664 ± 0.039	4.27
	OCTOPUS repro.	`qjl`	4.094	6.176 ± 0.032	11.545 ± 0.040	3.91
	HYPERQUANT	`qjl`	3.758	6.170 ± 0.025	11.526 ± 0.035	4.26
	HYPERQUANT	`none`	3.580	6.214 ± 0.000	11.495 ± 0.000	4.47
	HYPERQUANT, rate-matched	`qjl`	4.105	6.117 ± 0.019	11.443 ± 0.020	3.90
	HYPERQUANT, rate-matched	`none`	3.888	6.123 ± 0.000	11.421 ± 0.000	4.11
2	TurboQuant repro.	`qjl`	2.793	6.975 ± 0.107	12.642 ± 0.120	5.73
	OCTOPUS repro.	`qjl`	3.129	6.562 ± 0.087	12.055 ± 0.064	5.11
	HYPERQUANT	`qjl`	2.802	7.008 ± 0.082	12.913 ± 0.126	5.71
	HYPERQUANT	`none`	2.644	6.562 ± 0.000	12.297 ± 0.000	6.05
	HYPERQUANT, rate-matched	`qjl`	3.142	6.500 ± 0.085	12.073 ± 0.097	5.09
	HYPERQUANT, rate-matched	`none`	2.948	6.370 ± 0.000	11.780 ± 0.000	5.43

of it the window alone explains. Two smaller effects push the same way. Our OCTOPUS runs the 3×3 joint rounding its own specification designates as default, which is 6–7% better in MSE than the scalar variant its published table appears to use (Appendix D.15), and the shared accounting charges it an BF16 per-vector norm where its specification asks for `fp32`. Both make the baseline harder to beat than its published numbers, and we prefer that direction of error.

D.15 IS OUR OCTOPUS REPRODUCTION FAITHFUL?

A KV comparison that quotes a competitor's own table rather than reproducing it leaves the reader unable to separate codec quality from harness differences. This appendix reports the reproduction and, more importantly, the evidence that it is faithful, since an unfaithful reimplementation of a baseline is worse than an honest citation of it.

What we implemented. OCTOPUS (Boss et al., 2026) shares TurboQuant's rotation preconditioning but quantizes contiguous *triplets* of rotated coordinates jointly rather than one coordinate at a time. Following its Sections 3.1–3.5, our implementation (`hyperquant.octopus`) splits each cached vector into a magnitude and a direction, rotates the direction, cuts it into $n_{\text{tri}}=\lceil n/3 \rceil$ triplets (43 at n=128, the last zero-padded), splits each triplet into its norm ρ and its unit direction on S^2, folds that direction onto $[-1,1]^2$ with the octahedral map, and Lloyd-Max quantizes the two octahedral coordinates (shared codebook) and ρ under OCTOPUS's asymmetric $(b_{\text{dir}}, b_{\text{nrm}})=(b+1, b-1)$ bit split. The emitted code tuple is chosen by OCTOPUS's local 3×3 joint-rounding search rather than by three independent nearest-centroid rounds, with the norm code taken nearest to the achieved inner product $t^\top \hat{n}$ rather than to $\|t\|$. The rotation is supplied by the harness, so HYPERQUANT, TurboQuant, and OCTOPUS all see the identical one.

Check 1: does it hit OCTOPUS's own published distortion? OCTOPUS reports a purely synthetic reconstruction-fidelity table (isotropic Gaussian keys and queries at d=128, averaged over

Table 23: Every codec in its *windowless* configuration, which is how OCTOPUS's published table appears to have been produced, against the windowed numbers of Appendix Table 22. WikiText-2 $\Delta\%$, mean over 3 seeds windowless and the same 8 as Appendix Table 22 windowed, boundary-K protection retained in both columns. Our windowless OCTOPUS lands within 0.1 pp of its published 2-bit figure and within 1.6 pp at 3 and 4 bits, and our windowless TurboQuant within 2.2 pp at 3 and 4 bits, which is the closest external check on either reproduction we can make end-to-end; the one large residual, TurboQuant at 2 bits, is discussed in Section 4.4.2. Both HYPERQUANT rotations appear at every rate, so the matched-rotation comparisons of Section 4.4.2 can be repeated here: at b=2 the `qjl` arm that TurboQuant edges by 0.03 PPL under the window, within noise (+15.4 against +14.9), leads it without one (+49.6 against +77.2). The window helps every codec and helps most at the lowest rate, so we give it to all of them in the main comparison; HYPERQUANT leads on rate-matched quality either way.

nom. bits	codec	eff. bps (no win.)	no window	window 32	published
4	TurboQuant repro.	4.62	+3.0	+0.6	+3.1
	OCTOPUS repro.	4.97	+1.1	+0.3	+2.7
	HYPERQUANT (`qjl`)	4.63	+1.6	+0.2	–
	HYPERQUANT (`none`)	4.44	**+0.4**	+0.1	–
	HYPERQUANT, rate-matched (`qjl`)	4.98	+1.1	+0.1	–
	HYPERQUANT, rate-matched (`none`)	4.76	+0.9	**−0.1**	–
3	TurboQuant repro.	3.65	+10.8	+3.3	+8.6
	OCTOPUS repro.	4.00	+6.4	+1.7	+7.2
	HYPERQUANT (`qjl`)	3.66	+5.9	+1.6	–
	HYPERQUANT (`none`)	3.48	+5.8	+2.4	–
	HYPERQUANT, rate-matched (`qjl`)	4.01	**+4.1**	**+0.8**	–
	HYPERQUANT, rate-matched (`none`)	3.79	+4.7	+0.9	–
2	TurboQuant repro.	2.69	+77.2	+14.9	+63.0
	OCTOPUS repro.	3.03	+34.8	+8.1	+34.7
	HYPERQUANT (`qjl`)	2.70	+49.6	+15.4	–
	HYPERQUANT (`none`)	2.54	+48.7	+8.1	–
	HYPERQUANT, rate-matched (`qjl`)	3.04	**+20.6**	+7.1	–
	HYPERQUANT, rate-matched (`none`)	2.84	+23.5	**+4.9**	–

64 seeds) for itself *and* for the TurboQuant-MSE baseline. That gives six published numbers per bit width that depend on no model, no dataset, and none of this paper's own measurements, so it is a direct numerical target rather than a plausibility argument. Appendix Table 24 reproduces it.

		cosine ↑		MSE/coord. ↓		IP abs. err. ↓	
b	codec	ours	published	ours	published	ours	published
2	TurboQuant-MSE (ours)	0.9406	0.9406	0.1159	0.1161	3.056	3.054
	OCTOPUS, scalar rounding	0.9548	0.9547	0.0893	0.0897	2.680	2.682
	OCTOPUS, 3×3 joint	0.9579	–	0.0832	–	2.591	–
3	TurboQuant-MSE (ours)	0.9831	0.9831	0.0340	0.0340	1.654	1.650
	OCTOPUS, scalar rounding	0.9871	0.9871	0.0260	0.0260	1.446	1.444
	OCTOPUS, 3×3 joint	0.9880	–	0.0243	–	1.399	–
4	TurboQuant-MSE (ours)	0.9954	0.9954	0.0093	0.0094	0.865	0.866
	OCTOPUS, scalar rounding	0.9965	0.9965	0.0071	0.0071	0.755	0.753
	OCTOPUS, 3×3 joint	0.9967	–	0.0067	–	0.733	–

Table 24: Our reproductions against OCTOPUS's published synthetic fidelity table (Boss et al., 2026, Table 1): isotropic Gaussian keys and queries, d=128, 1024 keys and 16 queries per seed, 64 seeds. Both of our codecs reproduce the published numbers to the precision they are printed at. OCTOPUS's published row corresponds to its scalar-rounding encoder; the 3×3 joint rounding its Section 3.5 describes as the deployed default is a further 6.0–6.8% lower in MSE, and has no published counterpart.

Our TurboQuant reproduction matches all nine of OCTOPUS's TurboQuant-MSE entries, to $\leq 1\%$ on MSE and to every printed digit on cosine. This is a considerably stronger statement than Appendix D.14's asymptotic check, because it is an agreement with a number a *third party* measured for the method we are reproducing, at exactly the three bit widths the KV comparison deploys. It also cross-validates the two reproductions against each other: they share only the harness, so a harness-level error would have to move both in the same direction by the same amount.

Our OCTOPUS reproduction matches the published entries as well, and in doing so identifies which encoder variant produced them. Run with OCTOPUS's legacy scalar rounding (its candidate set reduced to the single seed index) we obtain 0.0893, 0.0260, and 0.0071 against published 0.0897, 0.0260, and 0.0071. Run with the 3×3 joint rounding that its Section 3.5 states is the deployed default, we obtain 6.0–6.8% lower MSE than published at every bit width. The consistent sign and magnitude of that offset, next to exact agreement on the scalar-rounding variant, is what one would expect if the published table was generated before joint rounding was switched on. We therefore run the joint-rounding variant everywhere below, which is the stronger of the two and the one OCTOPUS specifies, and note that this makes our OCTOPUS rows slightly better than OCTOPUS's own reported numbers would predict. That is what the end-to-end ablation shows too: in the windowless configuration its published table appears to describe, our OCTOPUS is a little ahead of the published figures at 3 and 4 bits and level with them at 2 (Appendix Table 23), the direction a 6–7% MSE improvement predicts.

Check 2: does its bit allocation land where OCTOPUS says? OCTOPUS derives an asymmetric split from a Lagrangian on the per-triplet squared error and reports three quantitative properties of the resulting landscape: sweeping $(b{+}\delta,\ b{-}\delta)$ around each uniform reference b, the minimum is at $\delta{=}{+}1$ for every b tested, that minimum is 31–41% below the uniform (b, b) reference, and $\delta{=}{+}2$ raises MSE by 44–73%. Appendix Table 25 is our sweep.

b	δ	$(b_{\mathrm{dir}}, b_{\mathrm{nrm}})$	MSE/coord.	vs. uniform	code bps
	-1	$(1, 3)$	0.45592	$+223\%$	1.680
2	0	$(2, 2)$	0.14100	–	2.016
	$+1$	$(3, 1)$	**0.08321**	$\mathbf{-41.0\%}$	2.352
	-2	$(1, 5)$	0.45041	$+1107\%$	2.352
	-1	$(2, 4)$	0.12737	$+241\%$	2.688
3	0	$(3, 3)$	0.03732	–	3.023
	$+1$	$(4, 2)$	**0.02431**	$\mathbf{-34.9\%}$	3.359
	$+2$	$(5, 1)$	0.05386	$+44.3\%$	3.695
	-2	$(2, 6)$	0.12631	$+1218\%$	3.359
	-1	$(3, 5)$	0.03318	$+246\%$	3.695
4	0	$(4, 4)$	0.00959	–	4.031
	$+1$	$(5, 3)$	**0.00667**	$\mathbf{-30.4\%}$	4.367
	$+2$	$(6, 2)$	0.01659	$+73.1\%$	4.703

Table 25: Reproduction of OCTOPUS's bit-allocation sweep (its Section 3.3), $d{=}128$. Splits needing fewer than one bit in a stream are omitted. The minimum sits at $\delta{=}{+}1$ at every b, at $-41.0/-34.9/-30.4\%$ against the uniform reference and $+44.3/+73.1\%$ at $\delta{=}{+}2$, against OCTOPUS's reported 31–41% and 44–73%. The last column shows why the sweep is not rate-neutral: $2b_{\mathrm{dir}}{+}b_{\mathrm{nrm}}$ grows with δ.

All three properties reproduce, two of them to the digit. That the $\delta{=}{+}2$ penalty comes out at $+44.3\%$ and $+73.1\%$ against a reported range of 44–73% is difficult to obtain by accident from an independent implementation, and it exercises the octahedral map, both codebooks, and the joint rounding together.

Check 3: the octahedral-coordinate marginal. OCTOPUS gives the marginal of one octahedral coordinate under the uniform prior on S^2 in closed form (its Eq. 7) but states that its implementation fits the direction codebook to empirical samples of $\mathrm{Oct}(n)$ instead. We did both. The closed form integrates to 1.000003 on our grid, and the codebooks fitted the two ways agree to $\leq 4.7 \times 10^{-4}$ per centroid at $b_{\mathrm{dir}} \in \{3, 4, 5\}$, a gap well inside the resolution of the histogram. We use the empirical fit, matching the specified implementation.

What the reproduction does not settle. Three fidelity gaps remain, and none of them is closed by the checks above. Our implementation follows OCTOPUS's published specification rather than its released Triton kernels, so any behavior of those kernels that departs from the paper's Algorithms 1 and 2 is not captured, and the scalar-versus-joint-rounding discrepancy above shows the two are not guaranteed to coincide. Our comparison quantizes K and V with the same codec, whereas OCTOPUS pairs a varying K codec with a fixed group-of-32 scalar V codec, so our OCTOPUS rows differ from its own on the value side. And we charge every codec one BF16 per-vector norm, where OCTOPUS specifies an `fp32` magnitude (0.125 bps more at n=128), so our accounting is marginally generous to OCTOPUS rather than to HYPERQUANT. What the reproduction does establish is that the codec we compare against reaches the distortion OCTOPUS itself publishes for it, at the bit widths in question, which is the property the comparison actually depends on.

E PROOF OF SUBTRACTIVE-DITHER UNBIASEDNESS

This appendix gives a self-contained proof that subtractive-dithered lattice quantization satisfies the Schuchman conditions and is hence *exactly* unbiased under inner products on every realization: for any deterministic query q and source x, the dithered reconstruction $\hat{x}$ satisfies $\mathbb{E}_U[\langle q,\, \hat{x}\rangle \mid x] = \langle q,\, x\rangle$ where the expectation is over the dither U alone. We then show the guarantee survives composition with the full HYPERQUANT KV pipeline and contrast it with the weaker approximate unbiasedness of QJL-without-dither.

E.1 SETUP AND NOTATION

Definition 1 (Lattice and Voronoi cell)**.** A lattice $\Lambda \subset \mathbb{R}^n$ is a discrete subgroup of $(\mathbb{R}^n, +)$ of full rank n, i.e., $\Lambda = \{\sum_{i=1}^n k_i b_i : k_i \in \mathbb{Z}\}$ for some $\mathbb{R}$-basis $b_1, \ldots, b_n$ of $\mathbb{R}^n$. The Voronoi cell of Λ at the origin is

$$\mathcal{V}(\Lambda) := \{x \in \mathbb{R}^n : \|x\| \le \|x - \lambda\| \text{ for all } \lambda \in \Lambda\}. \tag{2}$$

The Voronoi cell $\mathcal{V}(\Lambda)$ is a closed convex polytope. It is centrally symmetric, $\mathcal{V}(\Lambda) = -\mathcal{V}(\Lambda)$, because the defining inequalities (2) are invariant under $x \mapsto -x$ together with $\lambda \mapsto -\lambda$ (a bijection of Λ). The translates $\{\mathcal{V}(\Lambda) + \lambda : \lambda \in \Lambda\}$ tile $\mathbb{R}^n$, overlapping only on the measure-zero boundary $\partial\mathcal{V}(\Lambda)$.

Definition 2 (Fundamental domain)**.** A measurable set $D \subset \mathbb{R}^n$ is a fundamental domain for Λ if $\mathbb{R}^n = \bigsqcup_{\lambda\in\Lambda}(D + \lambda)$ up to sets of Lebesgue measure zero.

By the tiling property, $\mathcal{V}(\Lambda)$ is itself a fundamental domain. The covolume of Λ is $\mathrm{vol}(\mathcal{V}(\Lambda)) = |\det(b_1, \ldots, b_n)|$.

Definition 3 (Nearest-neighbor quantizer and mod-Λ projection)**.** Define

$$Q_\Lambda(y) := \arg\min_{\lambda\in\Lambda} \|y - \lambda\|, \qquad \pi_\Lambda(y) := y - Q_\Lambda(y),$$

with a fixed measurable tie-break rule on $\partial\mathcal{V}(\Lambda)$. The map π_Λ sends $y \in \mathbb{R}^n$ to the unique representative of $y + \Lambda$ in $\mathcal{V}(\Lambda)$ (uniqueness up to the boundary).

The map π_Λ has two properties we will use repeatedly:

(P1) *Range.* $\pi_\Lambda(\mathbb{R}^n) \subseteq \mathcal{V}(\Lambda)$.
(P2) *Lattice periodicity.* $\pi_\Lambda(y + \lambda) = \pi_\Lambda(y)$ for all $\lambda \in \Lambda$ and $y \in \mathbb{R}^n$, because $Q_\Lambda(y + \lambda) = Q_\Lambda(y) + \lambda$. Hence π_Λ descends to a well-defined map $\mathbb{R}^n/\Lambda \to \mathcal{V}(\Lambda)$.

E.2 THE MOD-Λ PUSHFORWARD IS UNIFORM

Lemma 1. *Let $D \subset \mathbb{R}^n$ be a fundamental domain of Λ with finite positive measure. If $Y \sim \mathrm{Uniform}(D)$, then $\pi_\Lambda(Y) \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$.*

Proof. Let $A \subseteq \mathcal{V}(\Lambda)$ be measurable. We have $\Pr[\pi_\Lambda(Y) \in A] = \mathrm{vol}(\pi_\Lambda^{-1}(A) \cap D)/\mathrm{vol}(D)$. By (P2), $\pi_\Lambda^{-1}(A) = \bigsqcup_{\lambda\in\Lambda}(A + \lambda)$, which tiles $A + \Lambda$ exactly once when restricted to any fundamental domain D. Hence $\mathrm{vol}(\pi_\Lambda^{-1}(A) \cap D) = \mathrm{vol}(A)$, so $\Pr[\pi_\Lambda(Y) \in A] = \mathrm{vol}(A)/\mathrm{vol}(D) = \mathrm{vol}(A)/\mathrm{vol}(\mathcal{V}(\Lambda))$, the uniform-on-$\mathcal{V}(\Lambda)$ probability. □

Corollary 1 (Crypto Lemma). *Let $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$. For every deterministic $x \in \mathbb{R}^n$,*

$$\pi_\Lambda(x+U) \sim \mathrm{Uniform}(\mathcal{V}(\Lambda)), \quad \textit{independent of } x.$$

Proof. Since $\mathcal{V}(\Lambda)$ is a fundamental domain, so is the translate $x + \mathcal{V}(\Lambda)$. The variable $Y := x + U$ has distribution $\mathrm{Uniform}(x + \mathcal{V}(\Lambda))$, and Lemma 1 applies with $D = x + \mathcal{V}(\Lambda)$. □

E.3 Subtractive dither produces an unbiased estimator

Definition 4 (Dithered reconstruction). Fix $x \in \mathbb{R}^n$ and let $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$ be independent of any other randomness. The subtractive-dithered reconstruction of x is

$$\hat{x}(x;U) := Q_\Lambda(x+U) - U. \tag{3}$$

The quantization error is $e(x;U) := \hat{x}(x;U) - x$.

Theorem 1 (Schuchman). *With U and $\hat{x}$ as in Definition 4, for every $x \in \mathbb{R}^n$ the error $e(x;U)$ is distributed as $-U' \sim \mathrm{Uniform}(-\mathcal{V}(\Lambda))$, independent of x. In particular, by central symmetry, $e(x;U) \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$ as well, and*

$$\mathbb{E}_U[e(x;U) \mid x] = 0. \tag{4}$$

Proof. Expand the error:

$$e(x;U) = \hat{x}(x;U) - x = Q_\Lambda(x+U) - U - x = -((x+U) - Q_\Lambda(x+U)) = -\pi_\Lambda(x+U).$$

By the Crypto Lemma (Corollary 1), $\pi_\Lambda(x+U) \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$, so $e(x;U) = -\pi_\Lambda(x+U) \sim \mathrm{Uniform}(-\mathcal{V}(\Lambda))$ independent of x. The mean is zero because $\mathcal{V}(\Lambda)$ is centrally symmetric. □

Corollary 2 (Inner-product unbiasedness). *For any deterministic $q \in \mathbb{R}^n$ and any source $x \in \mathbb{R}^n$,*

$$\mathbb{E}_U[\langle q,\, \hat{x}(x;U)\rangle \mid x] = \langle q,\, x\rangle.$$

Proof. By linearity of expectation and Theorem 1, $\mathbb{E}_U[\langle q,\, \hat{x}\rangle \mid x] = \langle q,\, x\rangle + \langle q,\, \mathbb{E}_U[e \mid x]\rangle = \langle q,\, x\rangle + \langle q,\, 0\rangle = \langle q,\, x\rangle$. □

Corollary 3 (Variance bound). *With the same notation, and writing $\sigma_\mathcal{V}^2 := \frac{1}{n}\mathbb{E}_{U\sim\mathrm{Uniform}(\mathcal{V})}[\|U\|^2]$ for the per-coordinate second moment of the Voronoi cell,*

$$\mathrm{Var}_U(\langle q,\, \hat{x}\rangle \mid x) = q^\top \mathrm{Cov}_U(U) q \le \lambda_{\max}(\mathrm{Cov}_U(U))\|q\|^2.$$

If $\mathcal{V}(\Lambda)$ is isotropic, i.e., $\mathrm{Cov}(U) = \sigma_\mathcal{V}^2 I_n$, this becomes $\mathrm{Var}_U(\langle q,\, \hat{x}\rangle \mid x) = \sigma_\mathcal{V}^2\|q\|^2$.

Remark. For the lattices used in HyperQuant ($\mathbb{Z}$, A_2, D_4, E_8), the Voronoi cell is isotropic, the lattice's symmetry group acts irreducibly on $\mathbb{R}^n$, and by Schur's lemma any invariant rank-2 tensor is a scalar multiple of I_n. Hence $\mathrm{Cov}(U) = \sigma_\mathcal{V}^2 I_n$ exactly. Numerically, $\sigma_\mathcal{V}^2(\mathbb{Z}) = 1/12 = 0.0833$ and $\sigma_\mathcal{V}^2(E_8) \approx 0.287$ in our scaling.

E.4 Composition with the HyperQuant KV pipeline

Proposition 1. *Let R be any orthogonal matrix ($R^\top R = I$), let $\alpha > 0$ be the lattice's calibration scale, and let $\hat{x}$ be the HyperQuant KV reconstruction of x:*

$$\begin{aligned} s(x) &:= \frac{\alpha\sqrt{n}}{\|x\|} \cdot Rx, \\ \widehat{s} &:= Q_\Lambda(s(x) + U) - U, \\ \hat{x} &:= \frac{\|x\|}{\alpha\sqrt{n}} R^\top \widehat{s}. \end{aligned}$$

Then $\mathbb{E}_U[\hat{x} \mid x] = x$, and in particular $\mathbb{E}_U[\langle q,\, \hat{x}\rangle \mid x] = \langle q,\, x\rangle$ for every deterministic $q \in \mathbb{R}^n$.

Proof. Apply Theorem 1 in the lattice's coordinate system to $s := s(x)$: $\mathbb{E}_U[\widehat{s} \mid s] = s$. The post-quantization map $\widehat{s} \mapsto (\|x\|/(\alpha\sqrt{n}))R^\top\widehat{s}$ is linear and depends only on x (not on U), so by linearity of conditional expectation, $\mathbb{E}_U[\hat{x} \mid x] = (\|x\|/(\alpha\sqrt{n}))R^\top s(x) = R^\top Rx = x$. Inner-product unbiasedness follows. □

Proposition 1 holds for any orthogonal R, deterministic identity, deterministic permutation, random Haar, random sign diagonal, including when R is itself random but *independent* of U, because the proof conditions on R and then averages.

E.5 Contrast: QJL alone is biased per-vector

The QJL-without-dither variant uses random rotation but no dither, so its reconstruction is

$$\hat{x}_{\mathrm{QJL}}(x; S) := S^\top Q_\Lambda(Sx), \qquad S \sim \mathrm{Uniform}(\mathrm{O}(n)), \tag{5}$$

with error $e_{\mathrm{QJL}}(x; S) = -S^\top \pi_\Lambda(Sx)$.

Proposition 2. *For every fixed realization $S = S_0 \in \mathrm{O}(n)$ and every fixed source $x \in \mathbb{R}^n$, the QJL error is a deterministic vector $-S_0^\top \pi_\Lambda(S_0 x)$, generally non-zero, so $\hat{x}_{\mathrm{QJL}}(x; S_0)$ is a biased estimator of x. There is no per-vector analog of Corollary 2 for QJL alone.*

Proof. Self-evident from (5): with S_0 fixed, neither side of the equation depends on any further randomness. □

Why the empirical QJL bias appears small. In a typical sweep test one averages $\frac{1}{N}\sum_{i=1}^{N}\langle q^{(i)}, e_{\mathrm{QJL}}(x^{(i)}; S_0)\rangle$ over many iid $(q^{(i)}, x^{(i)})$. Because q is independent of everything else and has zero mean, the sample average converges to $\langle \mathbb{E}q, \cdot\rangle = 0$. This is a consequence of $\mathbb{E}q = 0$, *not* of any QJL property; a non-rotated lattice quantizer passes the same empirical test. What QJL *does* provide is approximately isotropic error covariance: $S_0^\top \mathrm{Cov}_x(\pi_\Lambda(S_0 x))S_0$ is close to a scalar multiple of I_n when S_0 is a generic Haar draw. This is the genuine, non-vacuous benefit of the random rotation, but it is not the same property as unbiasedness.

E.6 Practical sampler

The proof requires $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$. The implementation uses the "mod-Λ trick":

$$U_{\mathrm{cube}} \sim \mathrm{Uniform}([-2, 2)^n), \quad U := \pi_\Lambda(U_{\mathrm{cube}}).$$

This is exact iff $[-2, 2)^n$ is itself a fundamental domain of Λ (Lemma 1). For E_8 the cube $[-2, 2)^8$ is not a union of Λ-translates of $\mathcal{V}(\Lambda)$, so the projection is only *approximately* uniform. A moment check alone ($\frac{1}{n}\mathbb{E}\|U\|^2$ against the analytical $\sigma^2_{\mathcal{V}}$, a 5% match) bounds a moment of the sampler, not the inner-product-unbiasedness guarantee Corollary 2 actually makes, so we test that guarantee directly: for 4096 random (x, q) pairs at $b{\approx}4$ (E_8^{int}), we draw 64 independent dither realizations per pair and measure the mean of $\langle q, \hat{x}_{\mathrm{dith}}\rangle - \langle q, x\rangle$ across all 262144 (pair, draw) samples. At the deployed cube half-width of 2 the mean bias is -1.2×10^{-3} (95% CI $[-4.7, +2.3] \times 10^{-3}$), i.e. $\approx 0.01\%$ of a typical $\langle q, x\rangle$ and statistically indistinguishable from zero; widening the cube to half-width $3, 4, 6$ gives -1.3×10^{-4}, $+9.6 \times 10^{-4}$, and -1.1×10^{-3} respectively, all with 95% CIs containing zero and no trend with cube size, so the boundary-truncation error the wide cube is meant to control is not detectable at this sample size, well below the level a moment-match check alone could certify. This is stronger evidence for the guarantee than the sampler's shape strictly requires: an exact sampler need only draw from *any* fundamental domain of Λ, not specifically $\mathcal{V}(\Lambda)$ (Lemma 1 does not require the Voronoi shape), and the fundamental-parallelepiped variant below gives one such exact alternative if a future deployment needs a sampler with a closed-form uniformity guarantee rather than an empirically-validated one: sample $T_i \overset{\text{iid}}{\sim} \mathrm{Uniform}[0, 1)$ for $i = 1, \ldots, n$, form $Y = \sum T_i b_i$ where $\{b_i\}$ is the lattice basis, and project $U := \pi_\Lambda(Y)$; U is then uniform on the fundamental *parallelepiped* (not $\mathcal{V}(\Lambda)$'s own shape), which still suffices for Lemma 1 since any fundamental domain gives an exact pushforward. We did not implement this variant, since constructing an E_8 basis consistent with the fast decoder's coset representation is a nontrivial, easy-to-silently-mis-tile addition, and the direct empirical test above already closes the gap it was meant to address.

F Calibration: setting the operating point

HyperQuant exposes one user-facing knob, the target rate b in bits/scalar, and turns it into a concrete quantizer in two steps: choose the quantization SNR that yields b, then set the per-vector

scale α that realizes that SNR. The first step needs an empirical rate curve $b(\mathrm{SNR})$, because the Rice-coded rate has no closed form; the second is a closed-form formula in the lattice's Voronoi second moment. Both are built once, on synthetic iid-Gaussian data, and, crucially, apply unchanged to every weight and KV tensor in any model (Appendix F.3). This is what lets HYPERQUANT hit an *arbitrary* fractional rate, a property fixed-rate codebooks cannot match.

F.1 FROM A TARGET RATE TO AN SNR (EMPIRICAL)

The rate of the stripped, Rice-coded stream is the composition of the lattice's granular gain, the bit-stripping transform, an *integer*-parameter Rice coder, and the 8-bit clip, none of which admits a clean closed form at the operating points of interest. We therefore measure it. For each lattice we draw $N \sim 10^5$ iid-Gaussian tiles, quantize at a grid of SNRs (the closed-form α of Appendix F.2 sets each point), and record the *realized* Rice rate. Appendix Figure 10 is the result.

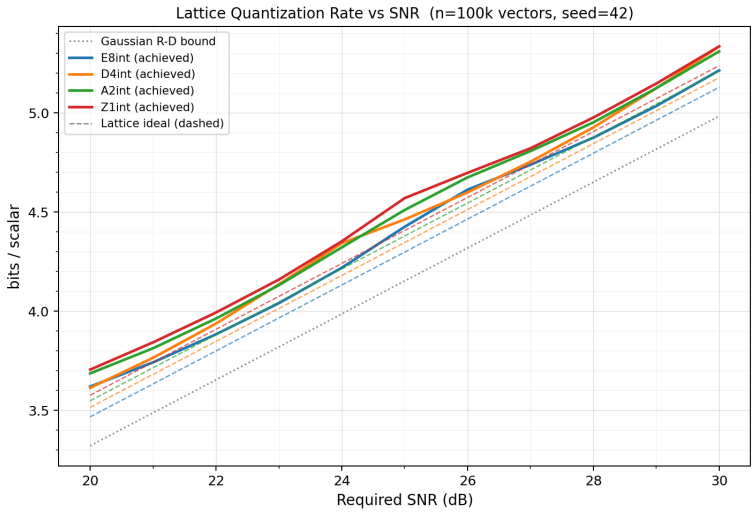


Figure 10: Empirical Rice rate versus target SNR on iid-Gaussian tiles (N=10^5, seed 42), for the four lattices. Solid: achieved bits/scalar; dashed: the lattice ideal $R_D + \frac{1}{2}\log_2(2\pi e\, G(\Lambda))$; dotted: the Gaussian rate–distortion bound $R_D = \frac{1}{2}\log_2 \mathrm{SNR}_{\mathrm{lin}}$. Each curve is smooth and monotone, so inverting it sends any target rate to a unique SNR; the achieved rate stays ≈ 0.1 bps above the lattice ideal throughout, ordering $E_8 < D_4 < A_2 < \mathbb{Z}$.

Two properties make this usable. *(i) Invertibility.* $b(\mathrm{SNR})$ is monotone increasing, so the implementation interpolates the tabulated curve to recover the unique SNR achieving any requested b; because the table stores the realized Rice rate (not an entropy estimate), selecting against it hits any target rate to within ~ 0.01 bps (Section 4.1 confirms this on-model). *(ii) Tightness.* The realized rate sits ≈ 0.1 bps above the lattice ideal. This gap is almost entirely the redundancy of the stateless, power-of-two Rice code over the symbols' *marginal* entropy; that marginal entropy already meets the lattice ideal (Appendix F.4), so the 0.1 bps reflects only the coder's simplicity; no residual lattice or inter-symbol inefficiency remains.

The per-lattice Rice structure is what the table is the rate *of*: E_8 uses a single parameter k_s (the coset bit folded into the combined symbol); D_4 and A_2 use two (k_s with $k_t = k_s - 1$ for D_4, and k_{t_y}, k_{n_x} for A_2); $\mathbb{Z}$ uses one. The halving identity $k_t = k_s - 1$ is checked at runtime (Section 3).

F.2 FROM AN SNR TO THE SCALE (CLOSED FORM)

Given the SNR, the scale is analytic. For $x \sim \mathcal{N}(0, I_N)$ the high-rate quantization error has mean square equal to the lattice's Voronoi second moment, $\mathrm{MSE}_{\mathrm{vor}} = G(\Lambda)\, N\, V_\Lambda^{2/N}$, while the signal power is $\mathbb{E}\|\alpha x\|^2 = \alpha^2 N$. Setting their ratio to the target $\mathrm{SNR}_{\mathrm{lin}} = 10^{\mathrm{SNR}/10}$ gives

$$\alpha(\mathrm{SNR}, \Lambda) \;=\; \sqrt{\frac{\mathrm{SNR}_{\mathrm{lin}}\, \mathrm{MSE}_{\mathrm{vor}}}{N}} \;=\; \sqrt{\mathrm{SNR}_{\mathrm{lin}}\; G(\Lambda)\, V_\Lambda^{2/N}}, \tag{6}$$

with V_Λ the covolume of the integer realization. No calibration data are needed for α; the only assumption is the high-rate approximation, which we verify: the empirical Voronoi second moment is within 5% of $\mathrm{MSE}_{\mathrm{vor}}$ for E_8, D_4, A_2 (within 20% for $\mathbb{Z}$, whose tiny cell is sensitive to bf16 rounding), and the realized SNR lands within ≈ 0.1 dB of the target across 20–30 dB.

The code fits a signed byte. Equation (6) also fixes the code magnitudes, and hence whether the stored integer coordinates fit INT8. After the per-tile RHT and ℓ_2 normalization each input scalar has unit variance (Appendix F.3); the quantizer rounds αu to the lattice, so each stored coordinate tracks αu_i, giving

$$Y_i \;\approx\; \mathcal{N}(0, \alpha^2), \tag{7}$$

the granular noise adding only $O(G(\Lambda))$ to the variance, negligible against α^2. An overflow ($|Y_i| > 127$) is therefore a $127/\alpha$-sigma tail event, with per-coordinate probability $\operatorname{erfc}\big(127/(\alpha\sqrt{2})\big)$.

Bit-stripping (Appendices C and F.4) only *shrinks* magnitudes: for E_8/D_4 the stored s-coordinates are $Y \gg 1$ and the parity coordinate is halved again, so the only coordinates kept un-shrunk are $\mathbb{Z}$'s scalars and A_2's n_x. The raw lattice integer is thus the binding case, and bounding it bounds the entropy-coded symbols a fortiori. The binding *rate* is the top of the sweep, 5 bps, where α, and with it the code magnitude, is largest; Appendix Table 26 evaluates (7) there.

Lattice	α	$127/\alpha$	$P(\lvert Y_i\rvert > 127)$	exp. overflows / $7{\times}10^9$
E_8	14.7	8.6σ	6×10^{-18}	4×10^{-8}
D_4	17.3	7.3σ	2×10^{-13}	1×10^{-3}
A_2 (n_x)	13.6	9.3σ	1×10^{-20}	7×10^{-11}
$\mathbb{Z}$	7.4	17σ	$<10^{-60}$	≈ 0

Table 26: Worst-case INT8 overflow at the top of the sweep (5 bps), the rate at which α, and hence the code magnitude, is largest. With $Y_i \approx \mathcal{N}(0, \alpha^2)$ an overflow is a $127/\alpha$-sigma event; the last column multiplies the per-coordinate tail by the $\sim 7 \times 10^9$ quantized weights of Llama-3.1-8B. The binding lattice is D_4 (largest α), still 7.3σ from the byte boundary.

Even the worst lattice, D_4, expects only $\sim 10^{-3}$ saturations across the whole model; the default E_8 sits 8.6σ out at $\sim 4 \times 10^{-8}$. And a saturation, when it occurs, is a clamp to ± 127 that perturbs one coordinate by less than the granular step, not a corruption. The analytic tail is moreover conservative: empirically (Appendix Figure 11) the largest coordinate sampled over the full 20–30 dB sweep is $\approx 3.4\sigma$ ($|Y| \approx 50$ for E_8), because the per-tile ℓ_2 normalization caps the maximum near $\sqrt{2 \ln 1024} \approx 3.7\sigma$, tighter than a free Gaussian. This justifies storing the raw code in one signed byte per scalar, the INT8 Tensor-Core format and the entropy-free fallback.

F.3 One calibration for all data

Both pieces above are built on iid-Gaussian tiles, yet HyperQuant applies them to real weights and KV activations with no per-tensor recalibration. The justification is the per-tile RHT followed by ℓ_2 normalization: the Hadamard rotation spreads each tile's energy across its coordinates, and normalization fixes its radius, so every tile is approximately an isotropic Gaussian on the sphere of radius $\alpha\sqrt{n}$, precisely the distribution the calibration was measured on. A single table and the closed form of Equation (6) therefore serve all tensors, layers, and models, which is what makes HyperQuant calibration-data-free.

F.4 Stripping is rate-optimal: marginal entropy meets the lattice ideal

A striking feature of the sweep is that the per-scalar *marginal* entropy of the stripped symbols ($\frac{1}{N}\sum_i H(s_i)$, reported as `scalar_entropy`) coincides with the lattice ideal $R_{\mathrm{ideal}} = R_D + \frac{1}{2}\log_2\big(2\pi e\, G(\Lambda)\big)$ to within $\sim 10^{-2}$ bps at every operating point and for every lattice (Appendix Figure 11, top, where the two curves overlie). This is not a coincidence: it is the composition of two classical high-resolution results with the design of the strip.

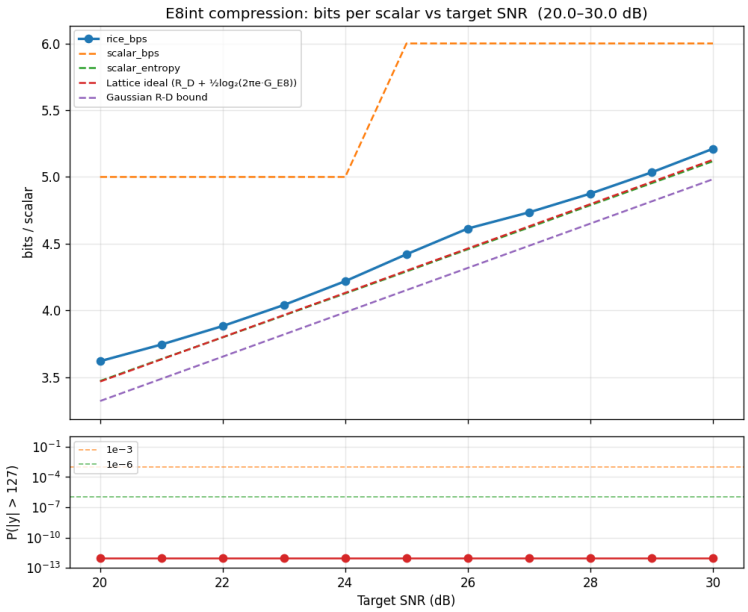


Figure 11: E_8 calibration detail. *Top:* achieved Rice rate tracks the lattice ideal within ≈ 0.1 bps over 20–30 dB (the dashed `scalar_bps` is the byte-aligned int8 fallback, which steps with the integer byte budget; the `scalar_entropy` curve, the marginal entropy of the stripped indices, lies essentially on the lattice ideal, cf. Appendix F.4). *Bottom:* sampled INT8 overflow rate $P(|c_i| > 127)$, which is 0 at every operating point (markers floored at 10^{-12} to render on the log axis); the largest coordinate seen is $\approx 3.4\sigma$, well inside the $\geq 7.3\sigma$ analytic margin of Appendix Table 26.

(i) The index entropy equals the lattice ideal. For a source X with finite differential entropy quantized by a lattice Λ at fine resolution, the index entropy obeys the high-resolution law

$$H\big(Q_\Lambda(X)\big) \;=\; h(X) \;-\; \log_2 \mathrm{vol}(\mathcal{V}(\Lambda)) \;+\; o(1), \tag{8}$$

the $o(1)$ vanishing as the cell shrinks (Gish & Pierce, 1968; Lookabaugh & Gray, 1989). With per-dimension distortion $D = G(\Lambda)\ \mathrm{vol}(\mathcal{V}(\Lambda))^{2/N}$ (Conway & Sloane, 1999, Ch. 3) and a white Gaussian source $X \sim \mathcal{N}(0, \sigma^2 I_N)$, for which $h(X)/N = \frac{1}{2}\log_2(2\pi e\,\sigma^2)$, dividing (8) by N gives

$$\frac{1}{N} H\big(Q_\Lambda(X)\big) \;\longrightarrow\; \underbrace{\tfrac{1}{2}\log_2 \frac{\sigma^2}{D}}_{R_D} \;+\; \tfrac{1}{2}\log_2\big(2\pi e\, G(\Lambda)\big) \;=\; R_{\mathrm{ideal}}. \tag{9}$$

The excess $\frac{1}{2}\log_2(2\pi e\, G(\Lambda))$ over the Gaussian rate–distortion bound R_D is the lattice's space-filling loss, exactly the redundancy of an entropy-coded dithered lattice quantizer above $R(D)$ (Zamir & Feder, 1992; Erez & Zamir, 2005; Zamir, 2014).

(ii) Stripping reduces the marginal sum to the joint entropy. The quantity `scalar_entropy` is not the joint entropy (9) but the per-scalar *sum of marginals*, $\frac{1}{N}\sum_i H(s_i)$. The strip Strip_Λ is a lossless bijection $\Lambda \leftrightarrow \mathbb{Z}^N$ (Appendix C), hence preserves the joint entropy, $H(s_1, \ldots, s_N) = H(Q_\Lambda(X))$, and by subadditivity

$$\frac{1}{N}\sum_i H(s_i) \;=\; \frac{1}{N} H\big(Q_\Lambda(X)\big) \;+\; \frac{C}{N}, \qquad C \;=\; \sum_i H(s_i) - H(s_1, \ldots, s_N) \;\geq\; 0, \tag{10}$$

with C the total correlation (multi-information) of the symbols (Cover & Thomas, 2006, Ch. 2). Two design choices send $C \to 0$. First, the strip is built to annihilate *exactly* the lattice's deterministic dependencies, the parity and coset constraints of Appendix C, the only exact couplings among the integer coordinates. Second, the residual *statistical* dependence vanishes in the high-rate limit: as the cell shrinks $Q_\Lambda(X) \to X$, so each stripped symbol converges to a scaled copy of an i.i.d.

Gaussian source coordinate ($s_i \to \alpha X_i/2$ for E_8^{int}, and likewise for the others) and the symbols become mutually independent. The subtractive dither of Corollary 1 makes this precise: it renders the quantization error independent of X (Zamir & Feder, 1992; Schuchman, 1964), removing the input-dependent part of the residual correlation at any rate. Hence $C/N \to 0$ and, combining (10) with (9), $\frac{1}{N}\sum_i H(s_i) \to R_{\text{ideal}}$.

Residual gap and consequence. The measured discrepancy is $\lesssim 0.02$ bps and changes sign across the sweep: the non-negative total correlation C/N pushes `scalar_entropy` *above* R_{ideal}, while finite-rate corrections to (8) and the ≈ 0.06 dB bf16 deficit between the requested SNR (at which R_{ideal} is evaluated) and the achieved SNR (at which the entropy is measured) push it *below*; both are $O(10^{-2})$ at these rates. The consequence justifies the strip-then-Rice design: because the stripped symbols are statistically near-independent and their marginal entropy already sits at the lattice ideal, a *memoryless* entropy coder is near rate-optimal: no context model or joint coder can recover more than the vanishing C/N. Stripping's rate-optimality is therefore a consequence of this construction, and it is what lets the stateless Rice coder give up only ~ 0.1 bps to the ideal.

G Scope and threats to validity

This appendix collects deployment- and evaluation-scope limitations that are not new experimental findings but that a reader assessing how far these results generalize should have in one place. Multi-seed significance, the dither sampler's empirical validation, the Gaussian-vs-Beta KV-marginal question, the same question for real weight tiles, and how much of the HIGGS margin is its dimension cap are addressed directly (Sections 4.4.1 and 4.4.2 and Appendices D.2 to D.4, D.10 and E.6), and the online-rotation cost, the TurboQuant and OCTOPUS reproductions, and the validation of both against published and asymptotic references are covered in Appendices D.8 and D.13 to D.15. The remaining items below – steady-state KV memory at long context, batch size, FlashAttention compatibility, a production-serving baseline, and long-context attention semantics – require either new benchmark integration (long-context evaluation, a live filled-cache measurement) or an unfamiliar production stack (vLLM/TensorRT-LLM), and are left for future work.

Pseudo-quantization harness. All KV-cache quality numbers (Sections 4.2 and 4.4.2) come from a harness that quantizes and immediately dequantizes to BF16 for the attention computation (Section 3.1); no experiment in this paper runs attention directly on the Rice-coded bitstream. The reported PPL and $\Delta\%$ numbers therefore measure the quantization noise correctly, but the resident-memory and throughput numbers for the KV path (Appendix Table 13) are computed from the bitstream size and a standalone decode kernel rather than measured on a live serving system in which the cache is stored compressed throughout generation.

Batch size. Every throughput and memory number in this paper is at batch size 1 (Appendix D.7). The Introduction's premise, that autoregressive decoding is memory-bandwidth-bound with idle Tensor Cores, holds at batch 1 but weakens at the batch sizes typical of production serving (8–256+), where arithmetic intensity rises and a fixed per-layer decode-kernel overhead is amortized differently. We have not measured how the memory and throughput trade-off changes at batch > 1.

Steady-state KV memory at long context. The $3.79\times$ KV compression figure (Appendix Table 13) is a per-vector rate computed from the encoding, distinct from a measured resident-memory footprint of a filled cache at a specific context length; Appendix Table 13's KV-only row shows resident memory is unchanged from baseline at model-load time precisely because the cache is empty then. We have not measured the actual HBM footprint of a filled cache at, e.g., 32K or 128K tokens, including the offset table, per-tile scales, and (when enabled) the QJL rotation matrices.

Online rotation cost for activations and the KV cache. The RHT's $O(n_{\text{tile}} \log n_{\text{tile}})$ butterfly is free for weights, which are rotated once offline (Section 3). For activations and the KV cache it must run online, once per tile per forward step. Section 3 states that it "folds into the preceding norm layer where possible"; here we isolate and measure the fold's latency and its coverage across layer types. The released integration's activation-side rotation (`_rht` in the weight-path linear layer, the same routine the KV path would use) costs $\approx 60\,\mu$s per call at batch-1 decode, essentially

independent of layer width (4096 or 14336), which means it is dominated by fixed dispatch/kernel-launch overhead in eager-mode PyTorch rather than by FLOPs (Appendix D.8). Summed naively over all 224 `nn.Linear` call sites in one Llama-3.1-8B forward (no fusion), this is ≈ 13 ms/token, comparable to or larger than the entire bf16-baseline per-token budget ($\approx$ 17–25 ms/token at the throughputs measured in Appendix D.7). That figure is a naive upper bound that assumes zero fusion coverage: tracing the released integration found no fold-into-norm-layer fusion on any weight-path or KV-path call site we inspected (the fold is asserted in Section 3 as something the design permits, and we did not find it in the code), but we did not instrument a per-site fold/no-fold ablation, so we cannot state what fraction of the 224 sites a fold would in fact cover. Two things follow. First, this cost is already inside the 1.85–$1.92\times$ end-to-end throughput gap reported in Section 4.3 and Appendix Table 13: every timed decode step calls the rotation unconditionally on the real activation, so it is counted there rather than being an additional cost on top (Appendix D.8 traces this in the code). Second, closing the fusion gap is still a more consequential open item for the throughput story than the Rice-decode kernel work in Section 3.1, since kernel-launch overhead of this size would swamp further decode-kernel speedups. For the KV path specifically, the rotation is per-(head, token) and sits on the autoregressive critical path, and the same unfused, per-call cost applies there as well; we have not separately isolated the KV-path call site.

FlashAttention / tiled-KV compatibility. Our KV hooks operate per-vector and are evaluated with standard (untiled) attention. Production serving typically uses FlashAttention-style kernels that load K/V in fixed-stride tiles directly from HBM; a variable-length Rice-coded cache is not naturally compatible with such tiled loads without a decode-to-BF16 step ahead of the attention kernel, which would consume part of the memory savings during attention itself. We have not measured or implemented this interaction.

No production-serving baseline. Our baselines (HIGGS, TurboQuant, OCTOPUS, NestQuant) are research codecs. We do not compare throughput or latency at matched quality against a deployed quantized serving stack (e.g., vLLM with AWQ/GPTQ, TensorRT-LLM FP8, or Marlin/CUTLASS INT8 kernels), so a practitioner cannot directly read off from this paper whether HYPERQUANT's memory advantage outweighs its kernel-level throughput regression relative to a well-tuned production system.

Gaussian calibration vs. the true KV marginal (resolved). HYPERQUANT calibrates every tile, including KV vectors, assuming an isotropic Gaussian source (Appendix F). After per-vector ℓ_2 normalization and a Haar rotation, KV coordinates lie on a fixed-radius sphere and their marginals are exactly Beta$((n-1)/2, (n-1)/2)$-distributed at the head dimension used here (n=128), approaching Gaussian only as n grows; this is precisely the marginal that motivates TurboQuant's Beta-calibrated scalar quantizer, and it is a consequence of the rotation rather than an assumption about the data, so it can be tested without any model at all. We did so directly (Appendix D.2): quantizing exact samples from the true Beta marginal at the alpha calibrated for i.i.d. Gaussian input costs no measurable SNR at every raw rate we tested in the high-compression regime (2.0, 2.5, 3.0 bps), at both head dimensions (≤ 0.01 dB at n=64, ≤ 0.002 dB at n=128); we ran n=64 because it is the regime a marginal-aware quantizer would most favor, the sphere marginal being furthest from Gaussian there. A Beta-calibrated scale would therefore not be expected to improve HYPERQUANT's KV results, though Appendix D.2 keeps the corresponding hedge, that we do not read this as evidence the mismatch is irrelevant in general.

PPL as a proxy for attention semantics at long context. The dither unbiasedness proof (Appendix E) is a statement about inner products, the right primitive for the value projection, not directly about the softmax attention distribution: a per-vector-unbiased key still adds variance to $QK^\top$, and softmax is nonlinear in those logits. We measure quality via perplexity, which averages over many tokens and can mask localized degradation (e.g., on long-context retrieval). Our KV comparison uses a 4096-token context even though Qwen2.5-7B-Instruct-1M supports far longer contexts; we have not evaluated HYPERQUANT on a long-context retrieval benchmark (e.g., Needle-in-a-Haystack or RULER) at the context lengths where attention concentration, and any localized effect of KV bias correction, would be most visible. We also have not verified whether the pre-RoPE, per-head rotation composes consistently across grouped heads under GQA/MQA beyond the architectural argument in Section 3.1.